Building Robust Planning and Execution Systems for Virtual Worlds

Don M Dini, Michael van Lent, Paul Carpenter, Kumar Iyer

Institute for Creative Technologies
University of Southern California
13274 Fiji Way, Marina del Rey, CA 90292
{dini, vanlent, carpenter, iyer}@ict.usc.edu

Abstract

Planning and execution systems have been used in a wide variety of systems to create practical and successful automation. They have been used for everything from performing scientific research on the surface of Mars to controlling enemy characters in video games to performing military air campaign planning. After reviewing past work on these various planning and execution systems, we believe that most lack one or more key components contained in another system. To enable future researchers to build more complete systems, and avoid possible serious system failure, we identify the major technical problems any implementer of such a system would have to face. In addition we cite recent solutions to each of these technical problems. We limit our focus to planning and execution for virtual worlds and the unique problems faced therein.

Introduction

Planning and execution systems are a proven technology with a long history of creating robust, intelligent systems for reasoning both in the real and virtual worlds. At Jet Propulsion Laboratory, for example, planning and execution systems have been used extensively to control Mars Rovers, which help to perform scientific research on the Martian terrain (Estlin et. al. 2005). In addition planning and execution systems are being used in commercial games (Orkin 2005) as well as virtual worlds for research (Riedl 2005; Mateas and Stern 2003; Van Lent, et al. 2005; Gordon and Logan 2004; Blumberg and Galyean 1997).

Before proceeding, it is useful to define precisely what is meant by a *planning* and *execution system*. In the most general sense, a *planner* performs the following function: given a description of the agent(s) world, and a description of goal criteria, a planner provides an action or action sequence that when executed will lead to achieving stated goal criteria. It represents the thinking part, before any acting is done. Investigation in planning and acting in virtual worlds in particular holds many benefits not found in real-world systems. Aside from the immense commercial rewards involved in helping to create game systems, creating AI for virtual worlds has great benefits for scientific research as well. As pointed out in (Tambe et. al. 1995) for example, "Another potential benefit is that artificial agents can simplify and speed up experimentation by providing more control of behavior, repeatability of scenarios, and increased rate of simulation (i.e., faster than real-time simulation)."

Although there have been many approaches to creating intelligent systems for virtual worlds, we believe that most lack one or more key components contained in another system. For this reason, we attempt to identify the key technical problems that anyone wanting to create a planning and execution system for virtual worlds would have to solve. In doing so, we hope to enable future researchers to build more complete systems that are less prone to potential failure.

The motivation for this survey of combined planning and execution systems comes, in part, from ongoing research projects at the University of Southern California's Institute for Creative Technologies (ICT). At ICT there are four different research efforts which have or are utilizing some combination of automated planning and plan execution. These are the Intelligent Forces project (van Lent, et al. 2005), the Automated Story Director project (Riedl 2005), the Virtual Humans project (Rickel et al. 2002), and the Reflective Tutoring and Explainable AI project (Core et al. 2006). The requirements and survey presented in this paper is the result of an attempt to identify the shared needs of these four projects and explore the possibility of a common architecture that allows various planners and execution systems to be treated as pluggable modules and can support the needs of current and future research projects.

The remainder of this paper is organized as follows. In the first section, we list and briefly describe each of the main technical problems. In the next section, we go into more detail and discussion on each problem and cite significant solutions to each. Lastly, a summarization and concluding remarks are provided.

Brief Description of Key Technical Problems

After performing a comprehensive survey of planning and execution systems, and intelligent systems built for virtual worlds in particular, we have identified the following list of key technical problems involved in creating a planning and execution system for virtual worlds. Each one is important in the sense that if a designer were to not address it, their system would exhibit a noticeable failure of some kind, as made apparent below. For each problem, we provide a definition and brief discussion on why it holds special importance in virtual environments.

Uncertainty. Following the description in (Ghallab, Nau, and Traverso 2004), uncertainty in planning in general comes from one of three sources: *Non-determinism*, *partial observability*, and *extended goals*.

Non-determinism refers to stochastic actions. In classical planning (Fikes and Nilsson 1971), actions are represented as STRIPS style operators, with fixed effects. In reality, however, the outcomes of actions are not totally knowable beforehand. This is typically modeled by saying after performing an action, the state of the world is one of many possible states, each occurring with a particular probability.

Partial observability refers to the imprecision with which the agent knows what the state of the world is. Unlike in classical planning, the agent has only a probability with which he knows something is true about the world state. As a concrete example, imagine running around in a maze. The state of the world might be your present location (x and y coordinates), and the only data you have to go on is what your surroundings look like. The problem is that many places in the maze look exactly the same, or very similar when you are immersed in it. Looking at the world thus only gives you an estimation of your present location, not a unique determination.

Lastly, uncertainty can arise due to *extended goals*. As in classical planning, one typically specifies the goal by indicating one or more goal conditions as a series of atomic sentences, the problem then being to find a plan that achieves one or more of them. In many realistic problems, however, goal conditions are not quite as clear cut. Using an example from (Ghallab, Nau, and Traverso 2004), one's goal may be a guarantee that a mobile robot avoid dangerous areas, and to reach a given location if possible. Thus in this case there are two separate goals, each with a different "strength."

Dynamic Environments. "Dynamic environment" here simply means that an agent cannot assume that elements of the world state will be constant; there will be elements that are likely to be quickly and constantly changing. For example, as considered in (van den Berg, Ferguson, and Kuffner 2006), an agent may be trying to perform path planning around moving obstacles.

Dynamic environments deserve particular attention when considering virtual environments. In video games, for example, action is extremely fast paced. Consequently, agents must be able to plan based on what *will soon* be true about the world, not based just on what is currently true.

Distributed Plans and Teamwork. AI systems often need to control a team of agents to perform a joint task. For example, teamwork among agents is critical in domains such as RoboCup soccer and RoboCup Rescue (Kitano, and Tadokoro 2001), and multi-robot space exploration (Estlin, Gaines, Fisher, and Castano 2005). One must build into the execution system some facility for making sure the task is performed in a coordinated manner. Otherwise as has been illustrated elsewhere (Levesque, Cohen, and Nunes 1990; Grosz and Kraus 1996), the task may fail.

Responding to a Plan Made Invalid During Execution. Plan recovery is a general tool that is required by components solving the other problems.
This problem is a large part of what motivates having an execution and monitoring system put on top of a simple planner in the first place. After the planner component creates a plan, in most realistic domains the plan becomes unviable during mid-execution for a host of reasons. The problem then, is how to maintain a plan that remains relevant to the desired goal set in the context of a dynamic environment.

Sensing and Maintaining Environment Data. There is a general need for timely, accurate data about the environment in which agents are acting to be relayed to the planning and execution system. This is of course necessary to create accurate plans that are relevant to the world. There are a few problems that arise when managing world data in a virtual environment, however. For example, as pointed out in (Orkin 2005), simply sensing data about the world may be computationally intensive in a virtual environment. Now the agent has the problem of planning with insufficient/partial data. One either has to obtain expensive world data in a judicious way, or go on planning without it.

A second problem related to managing data in virtual environments is deciding on what is relevant. In real world scenarios, the agent is limited only to the data reported back from the sensor. However, in virtual environments, one has access to all data in internal data structures representing the entire virtual world. This can be such a vast amount of data that sifting through it becomes difficult.

Authorability. At a high level, *authorability* refers to ease of content creation. For the problem of creating intelligent agents for acting in some domain, the "content" one must create is of the agent's knowledge. For example, a planning agent using a STRIPS style planner, one must create a domain specification, library of operators, or a task hierarchy (when using hierarchical planning).

Although authorability is certainly a factor in real world planners, it is more of a factor in virtual worlds. This can be seen in the following way. Consider the design process for a video game. A non-AI expert game designer will have an idea for a scene with dramatic elements that they would like to create. There must therefore be a way for the game designer to author agent knowledge easily, but still remain faithful to his dramatic vision.

Constraints on Plan Content. Additional constraints on plans occur in virtual worlds because agents must fulfill *roles*. In other words, there are restrictions on what actions are valid for a given agent. To see why this is so, consider the following. An agent in a video game is most often a human-like creature. Thus, any plan that is generated for it to carry out must adhere to things that only a human-like creature can believably do. (Tambe et. al 1995; Riedl and Young 2005). For example, an agent controlling a human character cannot walk through walls or jump over skyscrapers even if it would be conducive to a shorter or easier plan.

Solutions and Discussion of Main Problems

Dealing with Uncertainty

As mentioned above, uncertainty in planning can be resolved into three sources: Non-deterministic actions, partial observability, and extended goals. We will now consider notable solutions to each.

Non-determinism. A naïve response to non-deterministic actions would be to pretend that actions are still deterministic. That is, the effects of each action in its specification could be written as the most likely outcome. Then, if during execution, an action produces a non-predicted outcome, the agent could perform a recovery with conditional planning methods (Draper et al., 1994) or replanning (Wilkins 1990). If the environment is too non-deterministic, however, then recovery operations start to become expensive, and non-determinism becomes harder and harder to ignore.

One successful method for reasoning about stochastic actions include decision theoretic methods such as the Markov Decision Process (MDP) framework (Kaelbling, Littman, and Cassandra 1995), which is built from the ground up to reason about uncertain actions.

Additionally, classical planners and neo-classical planners have been extended to reason about uncertain actions. For example, there are probabilistic graph-planners and satisfiability planners, a survey of which appear in (Ghallab, Nau, and Traverso 2004)

Partial Observability. As mentioned above, partial observability of an environment means that an agent cannot uniquely determine the state of the world at any given moment. Instead, the agent maintains his best guess as to what state he is in. That is, a probability distribution over the set of states of the world is maintained (Ghallab, Nau, and Traverso 2004). This makes planning much more difficult, as now the state space, spanning the set of probability distributions, has become infinite.

One framework that has received an enormous amount of attention for solving partially observable problems is the Partially Observable MDP or POMDP framework (Kaelbling, Littman, and Cassandra 1995). The massive body of POMDP literature has advanced greatly, making computation of policies much easier, although still difficult in general.

Partial observability might arise in virtual environments for several reasons. Chief among them is the potential cost of performing sensing actions. Evaluating the precondition of an action is not simply the result of reading a robotic sensor. As (Orkin 2005) points out, determining the truth value of a precondition could be computationally expensive. If available resources are insufficient for evaluating a given precondition, then methods for planning with partial knowledge must be used.

Extended Goals. The general "extended goals" problem, as described above, is to find a way to reason about goals that have degrees of desirability. The need to support this in virtual environments is described in greater detail in the section "Dealing with a Highly Dynamic Environment." As (Ghallab, Nau, and Traverso 2004) describes, however, extended goals is a form of uncertainty because it is meant to capture the non-deterministic notion of having a goal that the agent is merely "trying" to achieve, rather than requiring the goal be performed.

This form of goal can be handled using utility functions and decision theory (Kaelbling, Littman, and Cassandra 1995). The formulation in this framework is there is now a "utility" associated with performing an action in a given state. The utility is calculated from the degree to which doing an action will lead the agent to more utility later on. The planning problem in this framework then becomes to find a plan (policy) that maximizes overall utility.

Responding to Broken Plans

If domains were totally deterministic, fully observable, and were static, then the outcome of each action could be known before hand, and there would be no reason to monitor and maintain plans during execution. Because of uncertainties and dynamic elements mentioned above, such as an unknown human player or stochastic actions, one has to deal with the problem of plans that become unviable during mid-execution.

To deal with this problem, there is a multitude of techniques developed to maintain relevant plans. Such methods include sensorless planning (Erdmann and Mason 1988), replanning (Wilkins 1990), using heuristics to fix broken plans (Estlin et. al 2002), and iterative repair (Chien et. al 2000). All of these methods are relevant to planning and execution in virtual worlds as well.

In addition, if one is able to anticipate the ways that a plan can go wrong, a good plan recovery method is *Contingency Planning* (Draper et al., 1994). In Contingency Planning, one creates a plan offline as a response for all the contingencies that could arise during plan execution. When a contingency happens, the execution system can just load the cached response plan. As discussed in more detail in a later section, these contingencies are easier to find and express in virtual worlds. This technique has been successfully applied in (Riedl, Saretto, and Young 2003).

Approaches to Teamwork

As noted earlier, plan steps sent to the execution component can in general be joint tasks, assigned to a team of agents to perform jointly. If some method for coordinating the execution of this task among the agents is not used, then failure of the task can result (Levesque, Cohen, and Nunes 1990). Generic methods of teamwork (Levesque, Cohen, and Nunes 1990; Grosz and Kraus 1996) have been developed that have been built into several successful multi-agent systems (Tambe 1997; Jennings 1995). (Levesque, Cohen, and Nunes 1990) for example, identifies a few key rules, referred to in (Levesque, Cohen, and Nunes 1990) as a joint persistent goal, that, if built into an execution system, result in true coordination. Specifically, if agents x and y are performing joint task T, then first, they should signal each other to make sure they start T at the same time. Second, and most importantly, if one of the agents privately comes to believe that T is either impossible or already accomplished, then they must inform the other agent of this.

Several multi-agent control systems have been built based upon joint intentions theory (Tambe 1997; Jennings 1995). Notably, (Tambe 1997) successfully implemented a multi-agent system and added the ability to use decision-theory methods for communication to minimize the cost of communication among agents.

The above generic teamwork methods are required for synchronization and consistency of joint actions. They do not address what the actual content of the distributed plan is, but rather how a team must perform each step in order to remain coordinated. Actually deciding on how to split a larger task among a team of agents to create a distributed plan is solved by a host of decentralized planning methods. Notable methods include game theory (Emery-Montemerlo, et al. 2005), phrasing distributed planning as a constraint satisfaction problem (Yokoo and Hirayama 2000), and distributed POMDPs (Nair et al. 2003). Other work has focused on encoding a human's knowledge of a domain into strategies for a team to follow (Hoang, Lee-Urban, and Munoz-Avila 2005)

Dealing with a Highly Dynamic Environment

A rapidly and constantly changing environment motivates several requirements for agents in virtual worlds, as we will now see. Let us first consider the sources of dynamism particular to virtual environments.

Video Games in particular are clearly a very dynamic environment, simply because many games are filled with fast paced action. Another major source of an unpredictable environment, however, is human users. An agent's planning component may make a sound plan, but the unknown and uncontrollable user may, through his actions, break a causal link making the plan invalid (Riedl 2005; Riedl, Saretto, and Young 2003).

The classical planning agent would solve this problem by creating a complete plan, and then executing it while holding its eyes closed. To account for a dynamic environment, a simple extension to this is to replan whenever the agent's plan goes off track. This is problematic, however, as replanning can be just as costly as planning (Jonsson and Backstrom 1995).

Monitoring Context. Instead, to deal with a dynamic environment, a planning and execution system must monitor the moment to moment context in which it acts (Geib 1994). As Geib points out, there are two successful methods for doing this. One is to precompute a policy offline, which means storing a cached response to each possible world state. This is the approach taken in (PO)MDP literature. Although this allows for planning with context and provides very fast response time, it suffers from two problems. First, dealing with any realistic problem requires a (possibly unacceptably) vast amount of offline computation. Second, the policy does not change during run time. This might be unsatisfactory for some problems.

The second method for performing efficient planning while monitoring context is interweaving the process of planning and execution. This is called incremental planning, or continuous planning (Jonsson and Backstrom 1995). In Incremental Planning, an agent produces only a *prefix* of a plan. While that is being executed, the agent can monitor any changes to the world and plan its next few actions. This method has been successfully applied in several domains, such as the avoidance of moving obstacles (van den Berg, Ferguson, and Kuffner 2006).

Further, it is more possible to examine context in virtual worlds than in the real world. This is because more data is available to the agent. Because the agent is operating in a virtual world, the internal representation of the environment is accessible. Unlike relying entirely on sensors while operating in the real world, any aspect of the virtual world that is stored in an internal data structure is accessible. Thus, where an ordinary sensor might simply tell you that a necessary precondition for the next step in the plan is no longer true, access to the world representation allows you investigate how that precondition became untrue.

Supporting Multiple Goals. A dynamic environment also motivates the requirement to support pursuing multiple goals at once, as noted in (Gordon and Logan 2004; Tambe et. al 1995). (Gordon and Logan 2004) gives an illustrating example. An agent in a Capture the Flag session in Epic's Unreal Tournament (UT) might be low on health, and so will adopt the goal of replenishing health. The problem is, if the health pack has been taken, and if the agent can only pursue one goal at a time, then it will single-mindedly wait for the health pack to reappear. It will be unprepared for imminent attacks from other players.

Some methods for pursuing multiple goals at once are described in (Gordon and Logan 2004; Mateas and Stern 2004). (Gordon and Logan 2004) describes a system that can dynamically generate new goals and dynamically assigns priority to goals based on the urgency of the agent's current situation. An "arbitrator" then decides which goal to pursue.

Sensing and Managing Data from a Virtual Environment

There is a very interesting problem in sensing the environment that is unique to virtual worlds. Because the environment is running as a simulation on a computer, obtaining data about the environment can sometimes amount to a costly computation (Orkin 2005). For example, determining if the precondition of an operator is true or not can require a costly path planning or ray intersection computation. The problem solved in (Orkin 2005) is how to perform the necessary calculations while not taking too many resources away from the other components of the game.

As (Orkin 2005) discusses, the solution to this problem is to not do too much calculation in a single frame of game play. Rather, one should intelligently spread out the calculation among many frames. As an example from the paper, if an enemy agent during game play discovers some threat, he stops and uses each frame to calculate the path to a location of tactical value, checking also if it is free of danger. The agent stops this process when a safe path is found.

The second problem identified above is how to deal with an overabundance of data. That is, when presented with a large amount of world data, the problem is to determine which data are relevant. At least one place where this problem is considered is in (Horvitz and Barry 1995).

Dealing with Constraints on Plan Content

As pointed out in (Tambe et. al 1995), most often an agent in a virtual system is playing the part of a human being, or human-like creature. This is especially so in video games. As mentioned above, there are consequently some constraints on what an agent in such an environment can do while maintaining believability.

Some past work on adding believability to agent behavior includes the Hap agent language (Loyall and Bates 1991). Hap was later extended to include most notably a mechanism for multi-agent coordination in the ABL (A Behavior Language) system (Mateas and Stern 2004). As (Mateas and Stern 2004) points out, the problem of achieving human-like believability for an agent contains to a large degree being able to perform multiple actions at once: "To achieve a non-trivial degree of life-likeness in such agents, they must possess the ability [to] perform several intelligent activities in parallel – for example, to gaze, speak, walk, use objects, gesture with their hands and convey facial expressions, all at the same time." The ability of the ABL system to coordinate all these activities to create believable behavior was illustrated impressively in the game Façade (Mateas and Stern 2003).

(Riedl and Young 2005; Mateas and Stern 2003) in contrast consider the problem of agent believability within the context of the field of story generation systems, in which one is concerned with the behavior of all agents in a virtual world for the purpose of relaying a narrative. The point discussed in (Riedl and Young 2005) is, when an agent's actions are the result of a planning system, believability is often harmed because the planner is only concerned with assembling actions to reach the desired goal. It is not concerned with the question of if those actions are role-appropriate for the agent.

An interesting consequence of requiring plan-appropriate behavior is the segmenting of players into roles. Specifically, different agents within a virtual world occupy different roles, and so have different sets of actions that are acceptable for them (e.g. a dragon can breath fire and fly, a human-like character cannot). As a result of this, one must consider how to coordinate *heterogeneous agents*. The problem to solve in this case is, given an unpredictable collection of agents, each with different abilities, how to coordinate them to solve a given task. One notable method for solving this problem is (Tambe et al. 2000) in which heterogeneous agents are coordinated by attaching proxies to them.

Authorability

Planning and execution systems require some method of authoring the knowledge for that domain. Any facility for making this creation process easier only adds to their utility. This type of authorability has unfortunately not received a great deal of attention in the AI community. In the field of virtual agents, some prior work has been done by (Robertson and Good 2005) where tools for creating interactive stories are discussed.

In addition to the above considerations, intelligent agents acting in virtual worlds are also subject to additional authorability constraints. Specifically, as mentioned above, agents acting in interactive story systems often motivate a need for a story director to exert a sudden control over an agent (Blumberg and Galyean 1997).

(Blumberg and Galyean) resolve agent control into four "levels": 1. specific motor action command, 2. behavior command, 3. motivation command (e.g. "you're cold"), and 4. manipulation of environment. To support these four levels of control, (Blumberg and Galyean 1997) create a system with three types of external run-time control. First, a director can adjust a "motivational variable," effectively changing a character's behavior at a more abstract level. Second, a director can change the condition on which a behavior activates, referred to as a "release mechanism." (Blumberg and Galyean 1997) give the example of initiating a dog's "marking" behavior. It may initially be set to activate upon seeing a fire hydrant. One could instead change the condition for this firing to be a human's pant-leg, if desired. The third available method is simply to allow the director to force execution of an action or behavior.

Summary and Future Work

Although many planning and execution systems have been made and used in a variety of circumstances, we believe that most systems lack one or more key components contained in another. To help make sure that systems built in the future are more complete, and do not lack a critical feature resulting in possible system failure, we provide in this paper a list of the key technical challenges involved in making a planning and execution system. However, we limit our focus to the particular problems involved in building systems for virtual worlds. In addition, we point out several significant solutions to these problems in recent literature. Our next focus is to investigate how to most efficiently build a planning and execution system containing each of these requirements. We will then implement our design and examine its performance in various virtual domains.

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

Blumberg, B., and Galyean, T. 1997. Multi-level control for animated autonomous agents: Do the right thing... oh, not that... In Trappl, R., and Petta, P., eds., Creating Personalities for Synthetic Actors. Springer-Verlag Lecture Notes in Artificial Intelligence.

Blythe, J. 1999. "Decision-Theoretic Planning", AI Magazine, Volume 20, Number 2, Summer.

Chien, S and Knight, R. and Stechert, A. and Sherwood, R. and Radibeau, G. 2000. Using Iterative Repair to Improve Responsiveness of Planning and Scheduling. In Proc. of AIPS. Breckenridje, CO, USA.

Core, M., Lane, H. C., van Lent, M., Gomboc, D., Solomon, S., and Rosenberg, M. 2006. Building Explainable Artificial Intelligence Systems. Accepted to appear in IAAI.

Draper, D., Hanks, S., and Weld, D. 1994. Probabilistic planning with information gathering and contingent execution. In Proceedings of ICAPS.
Emery-Montemerlo, R., Gordon, G., Schneider, J., and Thrun, S. 2005. Game Theoretic Control for Robot Teams. In Proceedings of IEEE ICRA.

Erdmann, M. A. and Mason, M. 1988. An exploration of sensorless manipulation. IEEE Journal of Robotics and Automation, 4(4), 369-379.

Estlin, T. and Fisher, F. and Gaines, D., and Chouinard, C. and Schaffer, S. 2002. Continuous Planning and Execution for an Autonomous Rover. In Proc. 3rd Intl. NASA Workshop on Planning and Scheduling for Space, Houston, TX.

Estlin, T.; Gaines, D.; Chounard, C.; Fisher, F.; Castano, R.; Judd, M.; Anderson, R.; and Nesnas, I. 2005a. Enabling autonomous rover science through dynamic planning and scheduling. In Proceedings of *IEEE Aerospace.*

Estlin, T., Gaines, D., Fisher, F., and Castano, R. 2005b. Coordinating Multiple Rovers with Interdependent Science Objectives. In Proceedings of AAMAS.

Fikes, R. and Nilsson, N. 1971. A new approach to the application of theorem proving to problem solving. Artificial Intelligence, 2:189-208.

Geib, C. 1994. The Intentional Planning System: Itplans. In Proceedings of AIPSC.

Gerevini, A. and Serina, I. 2002. LPG: a Planner based on Local Search for Planning Graphs. In Proceedings of AIPS'02, AAAI Press, Toulouse, France.

Ghallab, M., Nau, D., and Traverso, P. 2004. *Automated Planning: theory and practice*. San Francisco, CA.: Morgan Kaufmann.

Gordon, E. and Logan, B. 2004. Game Over: You Have Been Beaten by a GRUE. AAAI Workshop Challenges in Game Artificial Intelligence

Graesser, A.C. and Lang, K.L. and Roberts, R.M. 1991. Question answering in the context of stories. Journal of Experimental Psychology: General, vol. 120.

Grosz, B. and Kraus, S. 1996. Collaborative Plans for Complex Group Action. In Artificial Intelligence. 86(2), pp. 269-357.

Hoang, H., Lee-Urban, S., and Munoz-Avila, H. 2005. Hierarchical Plan Representation for Encoding Strategic Game AI. In Proceedings of AIIDE.

Horvitz and Barry, M. 1995. Display of information for time-critical decision making. In P. Besnard and S. Hanks, editors, Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, pages 296--305. Morgan Kaufmann, San Francisco.

Jennings, N. 1995. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. AIJ, 75:195-240.

Jonsson, P. and Backstrom, C. 1995. Incremental planning. In Proceedings of 3rd European Workshop on Planning.

Kaelbling, L., Littman, M., and Cassandra, A. 1995. Planning and acting in partially observable stochastic domains. Technical report, Brown University, Providence RI.

Kitano, H; Tadokoro, S. 2001. RoboCup rescue: A grand challenge for multiagent and intelligent systems AI MAG. Vol. 22, no. 1, pp. 39-52.

Laird, J., Newell, A. and Rosenbloom, P. 1987. Soar: An architecture for general intelligence. Artificial Intelligence 33:1-64.

Levesque, H.; Cohen, P.; and Nunes, J. 1990. On acting together. In Proceedings of AAAI, 94--99.

Loyall, A.B., Bates, J. 1991. Hap: A reactive, adaptive architecture for agents. Technical Report, CMU-CS-91-147, Department of Computer Science, Carnegie Mellon University.

Mateas, M. and Stern, A. 2003. Facade: An Experiment in Building a Fully-Realized Interactive Drama. In Proceedings of GDC, Game Design Track.

Mateas, M. and Stern, A. 2004. A Behavior Language: Joint Action and Behavioral Idioms. Book Chapter in *Life-Like Characters, Tools, Affective Functions and Applications*, eds. H. Prendinger and M. Ishizuka, Springer.

Muñoz-Avila, H. & Fisher, T. 2004. Strategic Planning for Unreal Tournament Bots. Proceedings of AAAI-04 Workshop on Challenges on Game AI. AAAI Press.

Nair, R., Pynadath, M., Yokoo, M., Tambe, M., and Marsella, S. 2003. Taming Decentralized POMDPs: Towards efficient policy computation for multiagent settings. In Proceedings of IJCAI.

Nau, D., Au, T., Ilghami, O., Kuter, U., Murdock, J., Wu, D. and Yaman, F. 2003. SHOP2: An HTN Planning System, JAIR, 20:379-404.

Orkin, J. 2005. Agent Architecture Considerations for Real-Time Planning in Games, AIIDE 2005 Proceedings.

Rickel, J., Gratch, J., Hill, R., Marsella, S., Traum, D., and Swartout, W. 2002. Toward a New Generation of Virtual Humans for Interactive Experiences, IEEE Intelligent Systems, July/August 2002, pp. 32-38.

Riedl, M. and Saretto, C. J. and Young, R. M. 2003. Managing Interaction between users and agents in a multi-agent storytelling environment. In Proceedings of AAMAS.

Riedl, M. 2005. "Towards Integrating AI Story Controllers and Game Engines: Reconciling World State Representations." IJCAI Workshop on Reasoning, Representation and Learning in Computer Games.

Robertson, J. and Good, J. 2005. Adventure Author: An Authoring Tool for 3D Virtual Reality Story Construction. AIED Workshop on Narrative Learning Environments.

Tambe, M and Johnson, W. L. and Jones, R and Koss, F. and Laird, J. E. and Rosenbloom, P. S. and Schwamb, K. 1995. Intelligent agents for interactive simulation environments. AI Magazine, 16(1), Spring.

Tambe, M. 1997. Towards flexible teamwork. JAIR, 7:83-124.

Tambe, M, Pynadath, D., Chauvat, N., Das, A., and Kaminka, G. 2000. Adaptive agent integration architectures for heterogeneous team members. In Proceedings of ICMAS.

van Lent, M., Riedl, M., Carpenter, P., McAlinden, R., Brobst, P. 2005. Increasing Replayability with Deliberative and Reactive Planning. In Proceedings of AIIDE.

van den Berg, J., Ferguson, D., and Kuffner, J. 2006. Anytime Path Planning and Replanning in Dynamic Environments. In Proceedings of ICRA.

Wilkins, D. 1990. Can AI planners solve practical problems? Computational Intelligence, 6(4), 232-246.

Yokoo, M., Hirayama, K. 2000. Algorithms for Distributed Constraint Satisfaction: A Review. *Autonomous Agents and Multi-Agent Systems* 3:2 185-207.