Soar-RWA: Planning, Teamwork, and Intelligent Behavior for Synthetic Rotary Wing Aircraft

Randall W. Hill, Jr., Johnny Chen, Jonathan Gratch, Paul Rosenbloom, Milind Tambe

Information Sciences Institute and Computer Science Department
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292
{hill, chen, gratch, rosenbloom, tambe}@isi.edu

1. Abstract

We have constructed a team of intelligent agents that perform the tasks of an Army attack helicopter company and a Marine transport/escort combined team for a synthetic battlefield environment used for running large-scale military exercises. We have used the Soar integrated architecture to develop: (1) pilot agents for a company of helicopters, (2) a command agent that makes decisions and plans for the helicopter company, and (3) an approach to teamwork that enables the pilot agents to coordinate their activities in accomplishing the goals of the company. This case study describes the task domain and architecture of our application, as well as the benefits and lessons learned from applying AI technology to this domain.

2. Task Description

2.1 Background

Since 1983 the Defense Advanced Research Projects Agency (DARPA) has exerted a significant effort to develop a realistic, distributed, synthetic battlespace that could be used for training, mission planning and rehearsal, tactics and doctrine development, and weapon-system concept evaluation. The underlying Advanced Distributed Simulation (ADS) technology builds large-scale simulations from a set of independent simulators linked together in a network (DIS Steering Committee 1994). It is envisioned that a synthetic battlespace will make it cheaper and safer to conduct large scale military exercises than would be possible with live field units. One of the goals of DARPA's Synthetic Theater of War '97 (STOW-97) project was to field several thousand entities (i.e., synthetic individual combatants and vehicles) during one exercise, using a modified version of ModSAF (Modular Semi-Automated Forces)(Calder et al., 1993).

Simulation in an ADS environment is high-fidelity: actions are represented and resolved at the level of entities. The synthetic environment includes terrain, oceans, and environmental effects (e.g., weather), all of which affect the perception and actions of the entities.

It is very expensive to field thousands of human troops for a field exercise, but deploying thousands of entities in a large-scale synthetic theater of war is also a challenge. One of the current weaknesses of the synthetic forces in ModSAF is that they are only capable of a limited amount of autonomy. Entities can be tasked to perform low-level actions, e.g., move in formation along a route, and attack an objective, but they are unable to perform a complex mission without a human controller to intervene and guide them from task to task. Furthermore, even with human controllers, the behavior of the ModSAF entities often does not achieve the desired level. Consequently, running a large-scale simulation could potentially be very costly in terms of the number of human controllers that would be required, and the quality of the simulation may not reach the desired level of quality. Herein lies the motivation for applying artificial intelligence techniques to the development of entities in this domain.

This paper describes our efforts to address these challenges. This work is part of a multi-phase project by a consortium of researchers from the University of Southern California Information Sciences Institute (USC-ISI), University of Michigan (UM)[1], and Carnegie Mellon University (CMU). The first phase emphasized the development of individual Soar/IFOR (Intelligent Forces) agents for the fixed wing aircraft domain. During this phase we developed a system using the Soar agent architecture (Tambe et. al, 1995; Laird et. al, 1995). This has served as the foundation for the work described in this paper. The second phase of the project continued the development of the Soar/IFOR agents, with UM

[1] We wish to acknowledge the UM team for their role with USC-ISI in developing the FWA agent architecture: Karen Coulter, Randy Jones, Frank Koss, John Laird, and Paul Nielsen.

developing new missions and capabilities in the fixed wing aircraft domain and CMU continuing their focus on Natural Language Processing. At USC-ISI we shifted our focus to the helicopter domain, while also adding new techniques to facilitate the perception of groups of entities (Hill, 1998), teamwork (Tambe, 1996a, 1997) and command-level planning and decision making (Gratch, 1996). Here we describe how these new approaches help address the challenge of constructing intelligent synthetic forces. We begin by giving an overview of two types of missions we deployed at STOW-97: the Army deep attack mission and the Marine escort/transport mission.

2.2 Overview of the Attack Helicopter Company

An army attack helicopter company (AHC) has eight attack helicopters and pilots, and a company commander who plans the company's missions and makes high-level decisions for the company. There are three AHC's in an attack helicopter battalion, and we deployed all three companies to perform the deep attack mission at STOW-97. The battalion also has a commander, but we do not yet model this agent. Instead, a human represents the battalion commander: the battalion-level plans are generated by a human and sent in an operations order, which is the standard military message for communicating mission plans to subordinates.

In a typical mission the battalion commander sends an operations order to each of the AHC commanders. Each of the AHC commanders analyzes the operations order and plans the mission for their respective companies. Once the planning is complete and the company's plan has been approved, the AHC commander sends an operations order to the AHC pilot agents, who execute the mission. The mission usually involves following a route to a position deep in enemy territory, avoiding enemy forces when possible and suppressing them when contact is made, moving to a battle area, destroying enemy targets in an engagement area, and returning to home base once the mission's objectives have been achieved.

2.3 Overview of the Transport/Escort Mission

The Marine transport/escort mission is really a pair of missions—one Combat Assault Transport mission and one Armed Escort mission. The purpose of the composite transport/escort mission is to move dismounted infantry from a staging area to a pre-specified landing zone, which is often in enemy territory. For STOW-97 the mission was configured to lift a company-sized unit—company-level lifts consisted of an 8-4-4 package: 8 CH-46E transport helicopters, 4 CH-53E transport helicopters, and 4 AH-1W attack helicopters. The Marine helicopters are tasked via an exercise editor by a human operator. Once the synthetic pilots receive their tasking they behave autonomously for the duration of the mission—many of their behaviors require a high degree of coordination and teamwork.

2.4 Objectives

Our overall goal is twofold. First, we seek to develop intelligent agents capable of acting autonomously on the synthetic battlefield so as to reduce the amount of human control needed to run large-scale military exercises. Second, we want our agents to be as human-like in behavior as possible so that the quality of the simulation meets the high standards of the domain experts.

To meet these goals, we need to develop AHC pilot agents that can fly their helicopters and execute their missions in the context of a complex, dynamic, and uncertain synthetic-battlefield environment, and they need to be able to act autonomously for relatively long periods of time—hours to days at a time. In addition, the individual pilot agents within the company must be able to act as a team during the execution of a mission to achieve the company's goals.

With respect to the company command agent, we need to model decision-making from a broader perspective and over longer time scales than what is done by the individual pilot agents. Whereas the pilot agents' decision-making tends to be more reactive in nature, the commander must deliberate about alternate courses of action, project effects into the future, and detect harmful (or beneficial) interactions between subordinate units and enemy forces.

This paper is a case study on how we applied AI technology to the development of the agents in the attack helicopter company and the transport/escort groups. We begin by describing the application's design and implementation, giving an extended example of what tasks the agents in the Army AHC and Marine transport/escort mission perform. Next we describe the use of AI technology in our agents, starting with the basic agent architecture and describing the extensions to incorporate teamwork and planning. Following this, we describe how our synthetic pilots and commanders were used in preparation for and participation in the STOW-97 advanced concept technology demonstration. Finally, we briefly describe our current research in the virtual battlespace.

3. Application Description

3.1 Design and implementation

The overall architecture of the Army helicopter company is shown in Figure 1. The distributed interactive simulation environment is provided by ModSAF: each ModSAF application program runs on a different machine on the network. Each ModSAF simulates multiple entities (i.e., vehicles), which are multicast to the other ModSAF's on the network. While ModSAF provides the simulation of the vehicles, in our case the AH-64 Apache helicopter, we implemented the pilot agents who fly the helicopters and the company command agent in Soar (we describe Soar in the section on AI Technology).

The Soar helicopter pilot agents perceive and take actions in the synthetic environment via the Soar-ModSAF interface (SMI), which is a collection of ‘C’ programs that provide the agents access to the dynamics and state of their vehicles as well as providing simulated sensors for detecting other entities in the environment via vision. The agents control their aircraft, manipulate simulated lasers and weapons, sense the terrain, and communicate by radio via the SMI.

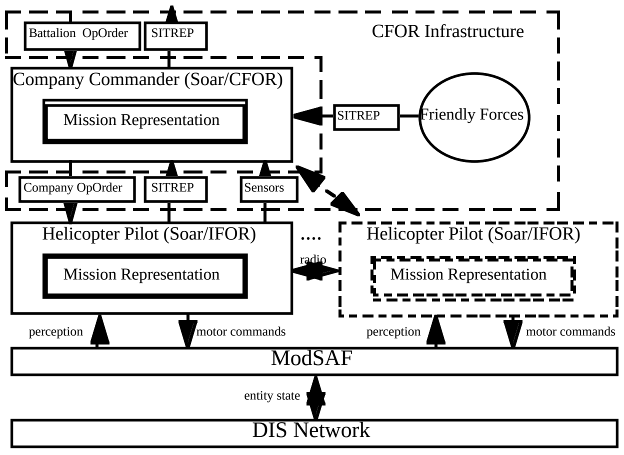

Figure 1: Architecture of the Attack Helicopter Company

Besides using simulated radios to communicate, the Soar helicopter pilot agents and the company commander agent also communicate via the Command and Control Communications System Interface Language (CCSIL), (Hartzog and Salisbury, 1996), a structured language that enables the agents to send standard military messages (e.g., operations orders or situation reports) to one another. CCSIL messages are delivered using the Command Forces (CFOR) Infrastructure software (Salisbury, 1995), which uses remote procedure calls to enable agents outside of ModSAF to communicate with agents in ModSAF vehicles. This is important since the Company Commander agent (see Figure 1) does not run in ModSAF but needs to be able to communicate with the Soar helicopter pilot agents that do. In addition, CCSIL provides a convenient way for humans to compose, send, and receive messages from ModSAF and CFOR agents.

The CFOR Infrastructure software also provides an interface to the sensors on ModSAF vehicles as well as providing a set of terrain reasoning functions that can be used by the commander agent in planning routes for tactical movements.

The helicopters used for the Marine transport/escort mission have the same architecture as the Army AHC, except that they do not use CCSIL nor does a synthetic commander plan their missions. Instead, we developed an exercise editor for specifying the key elements of the transport/escort mission, which were encoded into the mission representation used by the synthetic Marine pilots.

Subsequently, a similar tool—the Operations Order generation tool—was created to generate operations orders for the Army AHC deep attack mission. All that was required of the user was to create a graphical operations overlay in ModSAF, save it as a CCSIL overlay (i.e., the overlay was converted into the CCSIL overlay format), and then specify to the exercise editor the names of some key control measures. This greatly enhanced our ability to quickly task the Army AHC—none of the pre-planned Operations Orders were used since the enemy units were not located in the predicted positions.

3.2 Examples of how system is used

In developing the behaviors of the synthetic Army helicopter pilots, we focused on the primary mission of an attack helicopter company, the attack mission.[1] In a typical attack mission, the battalion commander—who in this case is a human—issues a plan, called an operations order, to each of the company commanders in the battalion. The operations order contains the battalion’s mission, what is known about the current friendly and enemy situations, and specific orders for each of the companies concerning what they are supposed to do as well as when and where they are supposed to do it. For example, the section of the operations order for Company A may state:

On order, ingress on route RED to Holding Area BRAVO, then move to Battle Position CAIRO and attack by fire to destroy enemy forces in Engagement Area HORNET[2].

Besides telling the companies their missions, the operations order also contains information about the locations of routes, objectives, phase lines, battle positions, engagement areas, and other control measures.

The company commander agent receives and analyzes the operations order, then plans a course of action to achieve the mission. During its initial planning the commander agent seeks to find a route to its assigned battle position that maximizes cover and concealment from the enemy force while minimizing the distance traveled. When the command agent has completed its initial planning, it encodes the plan as an operations order and backbriefs it to the battalion commander, who has the option of approving or disapproving it. Once the operations order is approved, it is sent to the individual pilot agents, who read it, identify their tasks and roles in the plan, and begin to execute the mission.

In a typical attack mission, the company organizes itself into two sections, a light team and a heavy team.

[1] Other potential Army missions include reconnaissance and security.

[2] Although this fragment is in English, the system actually uses a formal representation language called CCSIL.

The light team leads the company formation and performs the role of scouting the route. The heavy team follows the light team and overwatches their progress, providing firepower support when the light team is attacked. The teams take off from the home base and follow a route to a holding area, which is an intermediate point just prior to the company's battle position. Once in the holding area, the helicopters assigned the *scout* role move forward to the battle position, flying nap-of-the-earth to remain hidden from the enemy. The scouts reconnoiter the battle position to determine whether it is safe for the company to move forward and whether there are enemy targets in the engagement area. If enemy forces are observed in the engagement area, the scouts call the company forward to the battle position.

The company occupies the battle position by first moving to individually assigned firing positions that were selected by the company commander on the basis of how well they provide cover and concealment to the helicopter while also providing an optimal range to fire missiles at the enemy vehicles. While occupying their firing positions, the helicopter pilots first mask (hide) behind terrain features such as a hill or ridge to conceal themselves from the enemy, whenever possible. In cases where there is no cover and concealment, the helicopters position themselves at the maximum stand-off range of their Hellfire missile system. To engage a target, the helicopter unmasks, selects a target, fires several missiles, and quickly re-masks. In order to insure its own survivability, the helicopter will shift its position laterally while masked and unmask in another position.

Once the termination criteria for engaging targets has been met, the company re-groups and flies back to the home base. If they should encounter an unexpected enemy force en route to/from their objective, it is necessary to take "actions on contact," which involves immediately reacting to the situation in order to insure survival and modifying their current plans to bypass or destroy the enemy. In some cases it is desirable for the company to evade the enemy force and avoid contact altogether. In this instance the commander must re-plan the route to keep the company out of the enemy's weapon range, and, if possible, out of visual contact with the enemy. In other cases it is desirable to engage the enemy force to suppress or destroy it. In this instance, if the company can see the enemy and is out of the enemy's weapons range, the commander needs to re-plan the route so that the company can approach and engage the enemy unit from a position that provides cover and concealment.

In the case of the Marine transport/escort mission, the transport helicopters typically begin on a troop transport ship or in an assembly area, where the dismounted infantry embark the aircraft. The transport helicopters take off and rendezvous at a link-up point with the escort helicopters, which begin the mission on a separate ship. Once together, the escorts provide protective cover to the transports during the entire flight to and from the landing zone. While en route, two of the escorts fly ahead of the transports, while the other two follow the formation, orienting their flight on the center of mass of the transport helicopters. At the landing zone the escorts take up covering positions to protect the transports while they briefly land and debark the dismounted infantry. The escorts engage enemy units in the area of the landing zone or which otherwise threaten the mission. Once the debarkation is complete, the transports get back into formation and the escorts again assume their positions in the front and back of the transports and return to a pre-designated home base.

4. Use of AI Technology

4.1 Soar-based agent architecture

The pilot agents and the commander agent are built within Soar, a software architecture that is being developed as a basis for both an integrated intelligent system and a unified theory of human cognition (Rosenbloom, Laird & Newell, 1993; Newell, 1990). Soar provides the agents with support for knowledge representation, problem solving, reactivity, external interaction, and learning, (though our agents do not currently take advantage of Soar's learning capabilities), and it allows for the smooth integration of planning and reaction in decision-making (Pearson et al., 1993; Laird & Rosenbloom, 1990).

Tasks and goals are represented in Soar as *operators*. Operators perform the deliberate acts of the system. They can perform simple, primitive actions that modify the internal state and/or generate primitive external actions, or they can perform arbitrarily complex actions, such as executing a mission. The basic processing cycle is to repeatedly propose, select, and apply operators to a state. Operator selection, application, and termination are all dynamically determined by the system's knowledge and preferences, stored in the form of productions. Any changes in goals, states, and perceptions can cause these productions to fire.

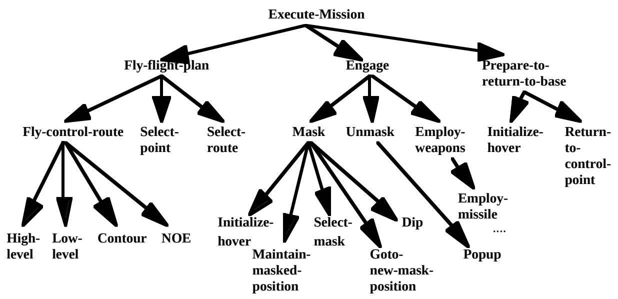

Figure 2: Part of the operator hierarchy for a pilot agent

In our application, agents interact with the Advanced Distributed Simulation (ADS) environment in real-time. This ability is facilitated by Soar's incorporation of perception within its decision loop. Decisions are informed by the current state of the world, as well as by rules that fire to interpret it. The perception system clusters visual entities based on proximity and summarizes this data in the agent's visual input, making it easier for the agent to reason about groups of individuals.

Figure 2 shows a portion of the operator hierarchy of the army helicopter agent. The Execute-mission operator decomposes into operators associated with different aspects of the attack mission. For example, the Fly-flight-plan operator decomposes into operators that enable the agent to fly a route from one location to another. Once the attack helicopter company has occupied the battle position, the Engage operator hierarchy is used for performing the tactics to destroy an enemy target while minimizing the danger to oneself. These operators serve to illustrate the kind of knowledge used and how it is organized for the attack helicopter domain.

4.2 Perception of Groups

In the course of developing the helicopter pilots a critical problem arose where the pilots would crash their aircraft when they perceived forty or more entities on the synthetic battlefield—the pilot could not cope with an excessive amount of perceptual input while simultaneously flying and performing other tasks. The problem of perceptual overload is not unique to the rotary wing aircraft (RWA) domain—it is a potential problem for every synthetic entity—but it is exacerbated in the RWA pilot by the fact that they fly low to the ground, making them susceptible to crashing, and yet their altitude makes it possible to perceive many entities at any given time.

Besides the problem of perceptual overload, it became apparent that the pilots needed to reason about large groups of other entities, which were being perceived as individuals. Seemingly simple tasks, such as following a group of other helicopters or flying around a group of enemy tanks, depended on reasoning about the locations and geometric relations to multiple entities. Such tasks are much simpler when a group of entities is treated as a group object rather than as individuals.

We addressed these problems by making a number of modifications to our pilot's perceptual system (Hill, 1998). To deal with perceptual overload, the perceptual system filters out unwanted visual objects (i.e., other entities). The pilot creates a visual filter by setting values for attributes such as slant-range, vehicle type, force-id, and group membership; the filter identifies entities the pilot wishes to perceive, and blocks everything that does not match the criteria. Using a visual filter saves the perceptual system from a lot of unneeded computation by limiting the pilot's visual attention to task-relevant entities.

To enable the pilot to understand the behavior of groups of others, the proximally located visual objects are clustered into group objects so that they are perceived as a group. The group objects provide an abstract representation both for reasoning and as a locus for visual attention—a visual filter can be set for a particular group. Furthermore, for most tasks, perceiving battlefield entities as groups is sufficient—it is not necessary to track every visual object all the time. Perceiving groups has greatly enhanced the ability of the agent to perform tasks such as escorting (Marine behavior) and avoiding enemy forces by flying around the group.

4.3 Teamwork

Teamwork in complex, dynamic domains, such as synthetic battlefields mandates highly flexible coordination and communication to surmount the uncertainties; e.g., dynamic changes in a team's goals, team members' unexpected inabilities to fulfill responsibilities (because they may crash, get shot down, run out of ammunition, or drop off the simulation network), and communication difficulties. Unfortunately, implemented multi-agent systems often rely on pre-planned, domain-specific coordination that fails to provide such flexibility (Jennings, 1995). First, it is difficult to anticipate and pre-plan for all possible coordination failures, given the complexity of the domain. Second, given domain specificity, reusability suffers—coordination has to be redesigned for each new domain.

To illustrate the need for teamwork, consider the following list of examples taken from our early work in developing the Army AHC.

- Upon reaching the holding area, the company waited, while a single scout started flying forward. Unfortunately, the scout unexpectedly crashed into a hillside; now, the rest of the company just waited indefinitely for the scout's scouting message.
- One pilot agent unexpectedly processed its initial orders before the others. It then flew towards the battle position, while its teammates were left behind at the assembly area.
- Only a scout made it to the holding area (all other helicopters crashed or got shot down); but the scout scouted the battle position anyway, and waited indefinitely for its non-existent company to move forward.
- When the initial orders unexpectedly failed to allocate the scouting role to team members, the company members waited indefinitely when they reached the holding point.
- Instructions sent by a pilot agent to some company members were lost, because they were sent while the members were busy with other tasks. Hence, these members were unable to select appropriate actions.
- While evading an enemy vehicle encountered en route, one helicopter pilot agent unexpectedly destroyed the vehicle via gunfire. However, this pilot agent did not inform the others; and thus an unnecessary, circuitous bypass route was planned.
- When all company members ran out of ammunition, the company failed to infer that their mission could not continue.

- Two separate companies of helicopters were accidentally allowed to use the same radio channels, leading to interference and loss of an initial message from one of the company commanders—its company hung indefinitely.

Each of these situations actually occurred in scenarios we ran to test the behaviors of the company during a deep attack mission. We initially attempted to remedy each of these problems by writing a specific coordination plan that would enable the pilots to cope with the particular situation, should it arise again. We soon discovered that once we had fixed one coordination problem, another one would arise, requiring another situation-specific coordination plan—there was no end in sight. Furthermore, the test scenarios were becoming increasingly more complex, making it more difficult to understand, replicate, and debug failures. There was no overarching framework that would anticipate and address teamwork failures.

A fundamental reason for these teamwork limitations is the current agent architectures. Architectures such as Soar (Tambe et. al, 1995), RAP (Firby, 1987), IRMA (Pollack, 1992), BB1 (Hayes-Roth et. al, 1995), and PRS (Rao et. al, 1993) facilitate an individual agent's flexible behaviors via mechanisms such as commitments and reactive plans. However, flexible individual behaviors, even if simultaneous and coordinated, do not sum up to teamwork. A common example provided is ordinary traffic, which although simultaneous and coordinated by traffic signs, is not teamwork. Indeed, theories of collaboration point to novel mental constructs as underlying teamwork, such as team goals, mutual beliefs and joint commitments (Grosz, 1996; Cohen and Levesque, 1991), absent in current agent architectures. Thus, agents cannot explicitly represent their team goals and plans, or flexibly reason about their communication/coordination responsibilities in teamwork; instead they rely on (problematic) pre-planned coordination.

In our work, we have integrated a set of teamwork capabilities within Soar; the combined system is called STEAM (Tambe, 1996a). STEAM is founded on the *joint intentions* theory (Cohen and Levesque, 1991). It enables explicit representation of team goals that expand out into goals and plans for individuals' roles in the team goal. In practice, to enable multiple team members to maintain a coherent view of their team's goals and plans, STEAM additionally incorporates (1) team synchronization to establish joint intentions; and (2) monitoring and repair capabilities. Unfortunately, communication in service of coherent teamwork can itself be a significant overhead or risk (e.g., radio silence in synthetic battlefields). Therefore, STEAM also integrates decision theoretic communication selectivity -- in the best interest of the team, agents may selectively avoid communication.

4.4 Planning

The demands of command-level decision making require a greater focus on deliberation than required by pilot agents. The commander must be proactive rather than reactive, anticipating the possible outcomes of future actions as well as potential interactions that might arise between actions. The greater focus on deliberation has led us to draw substantially from the classical planning literature in the course of command entity development. The command entity incorporates a hybrid of planning styles, incorporating hierarchical task-decomposition techniques (as in Tate, 1990) as well as partial-order planning approaches (as in Penberthy and Weld, 1992; Gratch, 1996). In this respect it closely resembles the IPEM planning architecture of Ambros-Ingerson and Steel (1988), with some significant enhancements. Task-decomposition planners plan by decomposing abstract tasks into a set of more concrete subtasks. Partial-order planners utilize the causal relationships between tasks to recognize various plan flaws such as missing tasks or ordering conflicts.

In the command agent, plans are represented by a graph structure known as a hierarchical task network (HTN). Nodes in the network correspond to tasks, and are represented as STRIPS-style action descriptions (Fikes, 1971). Tasks may be abstract or primitive. Abstract tasks may be decomposed into a partially ordered set of more specific tasks. Primitive tasks are those that may be directly executed without further decomposition.

Tasks in the network are connected by a variety of relations between them. Subtask relations define the basic hierarchical structure of the network. Ordering relations define the order in which tasks should be executed. Causal links and protection intervals are relations which represent the causal structure of the plan. (As in standard partial-order planners such as UCPOP, this causal structure facilitates reasoning about interactions across tasks.) Finally, dependency relations record information about how the plan was generated, for use in replanning. (Dependency relations are similar to the dependency graphs of Hayes (1975) and the verification structure of Kambhampati (1992).)

The three key activities performed by the command agent are plan generation, plan execution, and replanning. For plan generation, initially the planner is given an abstract plan (a battalion order). Typically the initial plan cannot be executed. It may contain non-primitive tasks or tasks may have unsatisfied preconditions. The flaws in the initial plan are addressed by non-deterministically selecting planning operations to resolve these flaws: fleshing out an abstract task (decomposition), finding existing tasks that satisfy preconditions (simple establishment), adding tasks (step addition), etc. These planning operations result in modification to the HTN and a complete plan is constructed through backtracking search.

Plan execution is facilitated through the use of a current world description. This is a variable-free set of literals that represents the sensed state of the world at the current time step. Execution operators allow tasks to be initiated or terminated. Only one task may be initiated or terminated on a give time step, but multiple tasks may be executing simultaneously. A task may be initiated if no unexecuted tasks precedes it and its preconditions unify with the current world description. A task may be terminated if its effects unify with the current world description. Currently, we do not have a general method for handling execution failure and rely on domain-specific procedures.

The current world description also facilitates the recognition of unexpected events. If there is a literal in the current world description that does not unify with any effect of any executed action, it is assumed to be the effect of some external event and is inserted into the task network.

Replanning may be triggered in response to unexpected events. If the effect of an unexpected event creates a flaw in the existing plan, the planner must modify the plan to address this new factor. The plan can be repaired in one of two ways: extending the plan (by adding new constraints or tasks), or retracting some portion of the plan and replanning. The former is the easiest approach, however it is not always possible to retain the original plan structure. If the unexpected event is incompatible with existing plan structure, this structure must be retracted. This process is facilitated through the use of the decision graph. This graph records the dependencies between planning decisions. In a method similar to Hayes' robot planner (1971), the command entity attempts to retract just those planning decisions directly affected by the new state of the world, retaining as much of the old plan as possible.

5. Application Use and Payoff

5.1 Operational Usage

The attack helicopter company and the Marine transport/escort mission has been deployed in numerous exercises and tests as a part of the STOW-97 program. They were first deployed in the Fall of 1995 for two events, first for a subsystem integration test in September and then for a major combined engineering demonstration called ED-1 in October. They have been deployed numerous times since then: Combined Test 1, in July, 1996; Combined Test 3, in October, 1996; and Combined Test 4, in December 1996, where we successfully ran three companies simultaneously for the Army deep attack mission. In the past year the Soar helicopter pilots were deployed in three functional system tests during the Summer and Fall, 1997, culminating in the STOW-97 Advanced Concept Technology Demonstration (ACTD) in October, 1997.

5.2 STOW-97 ACTD Results

A total of eight company-level missions were successfully run during the ACTD—three Army Attack missions and five Marine "Big Lifts" (i.e., combined Transport/Escort missions)—utilizing a total of 95 individual helicopter missions. Each of the three company-level Army attack missions were run with five AH-64 attack helicopters plus an automated company commander. Each helicopter had a set of ~1600 rules which defined its behavior, while each company commander had a set of ~900 rules to define its behavior.

The three Army deep attack missions were planned at the last minute, via the Exercise Editor and Operations Order generation tool that we provided, because the three pre-planned missions turned out to be operationally inappropriate (the opposing forces (OPFOR) were not in the locations and configurations anticipated). All three missions included engagements of significant bodies of OPFOR ground vehicles (and some air vehicles), and at least one included suppression of OPFOR en route. In all, 82 OPFOR entities were officially listed as killed, while three helicopters were officially lost. (The term "officially" is used here because the counts were officially adjusted from what actually occurred to offset inaccuracies in the simulation. In particular: (a) we shot down a MiG-21, but that was ruled to be too unlikely to have actually happened in the real world; (b) we engaged a group of OPFOR that proved to be invulnerable, so this group was reinstantiated and attritted to a level corresponding to the level of damage that might have been done by the helicopters should the OPFOR not have been invulnerable; (c) one company of helicopters was overrun and killed off by the invulnerable OPFOR, so they were reinstantiated with just one helicopter being listed as officially killed; and (d) three helicopters were killed at the Forward Assembly Area by a division of MiG-23s, but they were reinstantiated when the MiG attack was ruled to be bogus.)

During the ACTD we used two CPU's per attack helicopter company and thus two to three helicopters per machine. In exercises leading up to the ACTD we had been able to run a whole company on a single CPU, but during the dress rehearsal for the ACTD we found that a company-sized unit overloaded the machine. Thus we used six machines to run the aviation battalion plus two additional machines for the company commanders and a front-end GUI.

The five company-level Marine transport/escort missions—each of which in reality is composed of a pair of missions (one Combat Assault Transport mission and one Armed Escort mission)—comprised a battalion-level lift and two individual company-level lifts. Each company-level lift consisted of an 8-4-4 package; i.e., 8 CH-46E transport helicopters, 4 CH-53E transport helicopters, and 4 AH-1W attack helicopters. Four machines were used for each 8-4-4 package (and thus four helicopters per machine). Each helicopter in each mission had a set of ~1300 rules defining its behavior. The Marine missions were created via the previously described Marine RWA Exercise Editor. All missions required last minute modifications, which were also made through the Exercise Editor, due to changes in the situation.

The Marine missions ran successfully to completion, and resulted in successful deployment of Dismounted Infantry. During one of the missions the helicopters came under fire from a pair of OPFOR MiGs. The MiGs succeeded in destroying four of the helicopters (one AH-1W, two CH-46Es and one CH-53E) before the helicopters succeeded in shooting down the two MiGs.

Still, the remaining helicopters were able to reorganize and complete the remainder of the mission. During one other mission the helicopters came under fire from ground OPFOR in the Landing Zone (LZ). In this situation the helicopters suppressed the OPFOR both coming in and going out of the LZ.

5.3 Technology Evaluation

During the STOW ACTD we were able to test Soar plus the following multi-agent capabilities: (a) multi-agent perception; (b) teamwork reasoning; and (c) multi-agent planning. Soar worked without any known problems during the ACTD.

Multi-agent planning was used in the Army company commander. It made no known errors in fleshing out the company plans that were generated from overlays via the OpOrder generation tool. Multi-agent perception was used in both army and marine helicopters to group perceived entities and to filter out larger numbers of entities. It was critical in keeping the helicopters from being overwhelmed by the large numbers of entities that they perceived (and in allowing helicopters to perceive groups of entities). The army and marine helicopters used multi-agent perception to help them keep track of the other helicopters in their teams, which in turn helped support teamwork behavior. Teamwork reasoning enabled the helicopters to behave in a coordinated fashion even in the presence of unexpected changes, such as the loss of team members.

6. Current Research

Despite how well these capabilities all worked in the ACTD, limitations and problems do still exist. Some limitations showed up in the dress rehearsal but not in the ACTD, and others were worked around for the ACTD. We worked around the fact that the planner could not replan missions once a mission had started (by reinitializing entities and planning anew). We worked around that the perceptual system can still occasionally get overwhelmed (by reducing the number of helicopters per machine so that each pilot had more cycles to work with) and sometimes cannot see entities that are critical (by making the marine helicopters invulnerable to small arms fire from dismounted infantry, which they couldn't see). Situations in which companies could get stuck because one member would no longer cooperate in teamwork interactions, even though the team member was still alive, could have arisen during the ACTD, but did not. Our current research addresses these and other issues and limitations.

One of the insights we gained through our participation in STOW-97 was the importance of being able to understand groups of other agents. Our current research is focusing on how to extend the current pilot architecture to deal more effectively with interactions with groups of other entities.

6.1 Group Perception

At the perceptual level this means being able to handle the potential load induced by very large groups i.e., hundreds or thousands of entities. To this end we are investigating the use of more effective focus-of-attention strategies, a reduced (and more realistic) visual sensor field, and the imposition of hard limits in the perceptual system on how much processing can be done any given cycle. With respect to the perception of groups, we plan to extend the capabilities of the pilot to recognize formations and limited low-level aspects of behavior, based on shape and motion. In addition, we plan to experiment with representing groups at different resolutions. Currently groups are a fixed size—clusters have a 1 km radius—and the new approach would enable representing and reasoning about aggregates of groups of different sizes.

6.2 Group Understanding

At the pilot’s reasoning level, we are investigating how to understand groups of dissimilar others by enabling the pilot to “think like them.” Our previous research focused on how to model individual entities (Tambe, 1995), but it assumed that the entity being modeled had the same task hierarchy (i.e., knowledge and reasoning capabilities) as the pilot. This worked in a limited number of situations in the fixed wing aircraft (FWA) domain where pilots model other pilots during tactical maneuvers against one another. Our current research extends this approach: the pilot will model the thought processes of ground units; this means the pilot will incorporate models of dissimilar others and also scale-up to cover multiple entities.

6.3 Group Planning

We are currently focusing our planning research on the collaborative nature of the planning process. In STOW-97, we only automated the planning process at the company level. Each company planning agent independently developed its own plan (within the constraints of the battalion order) and this plan was either approved or disapproved by the (human) battalion commander. There was none of the give and take (e.g., negotiating over resources) that arises in real-life group planning situations. To address this issue we have introduced a battalion-level planning agent and are exploring ways of modeling methods of collaboration. As a first approximation we are considering an iterative approach to group planning. Initially the battalion commander generates high-level plans for each of its companies, and instructs them to plan out the details. These company plans are backbriefed to the battalion planning agent, who attempts to integrate them into a composite plan. This integration may fail, or suggest new opportunities, in which case the battalion agent communicates new constraints that the company plans must satisfy. This causes the company agents to modify their plans, and the process continues until a consensus is achieved.

6.4 Learning and Emotions

We are beginning research in two other interesting areas: learning and emotions. With respect to learning, the goal is to enable the pilots to improve their understanding of groups from experience. This will involve investigating ways to capture classes of changes and situations in a domain-independent manner. With respect to emotions, we are investigating how to model the effects of fatigue and stress on the pilot's decision making. This research will go beyond writing rules that say how an emotion would affect a decision in a specific situation. Instead, we are considering what it means to represent emotion in the Soar architecture itself.

7. Acknowledgment

We wish to acknowledge the UM team for their role with USC-ISI in developing the FWA agent architecture: Karen Coulter, Randy Jones, Frank Koss, John Laird, and Paul Nielsen.

This research was supported by the Defense Advanced Research Projects Agency, Information Sciences Office, and the Naval Command and Ocean Surveillance Center, RDT&E division (NRaD), under contract N66001-95-C-6013.

An earlier version of this paper was presented at IAAI-97 (Hill et al., 1997.)

8. References

Ambros-Ingerson, J.A. and S. Steel, 1988. "Integrating Planning, Execution, and Monitoring," Proceedings of the National Conference on Artificial Intelligence, 1988, pp. 83-88.

Calder, R.B., J.E. Smith, A.J. Courtemanche, J.M.F. Mar, A.Z. Ceranowicz, 1993. “ModSAF Behavior Simulation and Control,” Proceedings of the Second Conference on Computer Generated Forces and Behavioral Representation, STRICOM-DMSO, July, 1993, pp. 347-356.

Cohen, P. R. and Levesque, H. J., 1991. "Confirmation and Joint Action," Proceedings of International Joint Conf. on Artificial Intelligence.

DeJong, G, and Mooney, R, "Explanation-based Learning: An Alternative View," Machine Learning 1, 2, 1986, ppp 145-176/

DIS Steering Committee. 1994. The DIS Vision: A Map to the Future of Distributed Simulations, Technical Report, IST-SP-94-01, Institute for Simulation and Training, University of Central Florida.

Erol, K., Hendler, J., and Nau, D.S., "HTN Planning: complexity and expressivity," Proceedings of the National Conference on Artificial Intelligence, 194, pp. 1123-1128.

Fikes, R. E., and Nilsson, N. J., 1971. “STRIPS: a new approach to the application of theorem proving to problem solving,” AI Journal 2 (3/4) 1971.

Firby, J., 1987. “An investigation into reactive planning in complex domains," Proceedings of National Conf. on Artificial Intelligence.

Gratch, J.A., "Task-decomposition planning for command decision making," Proceedings of the Sixth Conference on Computer Generated Forces and Behavioral Representation, STRICOM-DMSO, July, 1996, pp. 37-45.

Grosz, B., 1996. “Collaborating systems," AI Mag, vol. 17.

Hartzog, S. M., Salisbury, M.R., "Command Forces (CFOR) Program Status Report," Proceedings of the Sixth Conference on Computer Generated Forces and Behavioral Representation, Orlando, Florida, July, 1996.

Hayes, P. J., 1975. “A representation for robot plans,” IJCAI-75, 181-188.

Hayes-Roth, B., Brownston, L. and Gen, R.V., 1995. “Multiagent collaobration in directed improvisation", In Proceedings of International Conf. on Multi-Agent Systems.

Hill, R., Chen, J., Gratch, G., Rosenbloom, P., and Tambe, M., 1997. “Intelligent Agents for the Synthetic Battlefield,” in AAAI-97/IAAI-97, pp. 1006-1012.

Hill, R. W., 1998. “Perception in a Multi-Agent Virtual World,” unpublished technical report, University of Southern California, Information Sciences Institute.

Jennings, N., 1995. “Controlling cooperative problem solving in industrial multi-agent systems using joint intentions," Artificial Intelligence, 75, pp. 195-240.

Jones, R.M., Laird, J.E., and Nielsen, P.E., 1996. “Moving Intelligent Automated Forces Into Theater-Level Scenarios,” in Proceedings of the Sixth Conference on Computer Generated Forces and Behavioral Representation, STRICOM-DMSO, July, 1996, pp. 113-117.

Kambhampati, S., 1992. “A validation-structure-based theory of plan modificaiton and reuse,” AI Journal 55: 193-258.

Laird, J.E., Johnson, W.L., Jones, R.M., Koss, F., Lehman, J.F., Nielsen, P.E., Rosenbloom, P.S., Rubinoff, R., Schwamb, K., Tambe, M., Van Dyke, J., van Lent, M., Wray, R.E., III, 1995. "Simulated Intelligent Forces for Air: The Soar/IFOR Project 1995," Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation, STRICOM-DMSO, pp. 27-36.

Laird, J.E. and Rosenbloom, P.S., "Integrating execution, planning, and learning in Soar for external environments," in Proceedings of the National Conference on Artificial Intelligence. Menlo Park, California: The AAAI Press, July, 1990.

Mitchell, T. M., Keller R. M., and Kedar-Cabelli, S. T, "Explanation-based Generalization: A Unifying View", Machine Learning 1 (1): 1986, pp. 47-80.

Newell, A., Unified Theories of Cognition. Harvard University Press, Cambridge, Massachusetts, 1990.

Pearson, D.J., Huffman, S.B., Willis, M.B., Laird, J.E., and Jones, R.M., "A symbolic solution to intelligent real-time control," IEEE Robotics and Autonomous Systems, 11:279-291.

Penberthy, J., Weld, D., 1992. "UCPOP: A sound, complete, partial order planner for ADL," In proc. 3rd Int. Conf. on principles of knowlede representation and reasoning, 103-114.

Pollack, M., 1992. "The uses of plans," Artificial Intelligence, volume 57.

Rao, A. S., Lucas, A., Morley, D. Selvestrel, M. and Murray, G., 1993. "Agent-oriented architecture for air-combat simulation", Technical Report 42, Australian AI Institute.

Rosenbloom, P.S., Laird, J.E., Newell, A., (Eds.) 1993. The Soar Papers: Research on Integrated Intelligence. Cambridge, MA: MIT Press.

Salisbury, M.R.,Booker, L.B., Seidel, D.W., Dahmann, J.S., "Implementation of Command Forces (CFOR) Simulation," Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation, 423-430, Orlando, Florida, May, 1995.

Tambe, M., Johnson, W.L., Jones, R.M., Koss, F., Laird, J.E., Rosenbloom, P.S., and Schwamb, K., 1995. "Intelligent agents for interactive simulation environments," AI Magazine, 16(1):15-39, Spring, 1995, AAAI.

Tambe, M., 1996a. "Teamwork in real-world, dynamic environments," In Proceedings of the International Conference on Multi-Agent Systems.

Tambe, M., 1996b. Tracking dynamic team activity. In Procceedings of the National Conference on Artificial Intelligence.

Tambe, M., 1995. "Recursive agent and agent-group tracking in a real-time dynamic environment." In Proceedings of the International Conference on Multi-Agent Systems (ICMAS).

Tate, A., 1990. "Generating project networks," In Allen, J.; Hendler, J; and Tate, A., editors, Readings in Planning. Morgan Kaufman. 162-170.

9. Authors' Biographies

Randall W. Hill, Jr. is a computer scientist at the University of Southern California Information Sciences Institute (USC-ISI) and a research assistant professor in the computer science department at USC. He received his B.S. degree from the United States Military Academy at West Point in 1978 and his M.S. and Ph.D. degrees in computer science from USC in 1987 and 1993, respectively. His research interests are in the areas of integrated intelligent systems, multi-agent systems, perception, and intelligent tutoring systems.

Johnny Chen was a programmer at the University of Southern California Information Sciences Institute (USC-ISI) at the time this work was performed. He currently resides in Dallas, Texas.

Jonathan Gratch is a computer scientist at the University of Southern California Information Sciences Institute (USC-ISI) and a research assistant professor in the computer science department at USC. He completed his undergraduate education in computer science at the University of Texas at Austin in 1986. He received his Ph.D. in 1995 from the University of Illinois in Urbana Champaign. His research interests are in the areas of planning, learning and decision theory.

Paul Rosenbloom is an associate professor of computer science at the University of Southern California and the deputy director of the Intelligent Systems Division at the Information Sciences Institute. He received his B.S. degree in mathematical sciences from Stanford University in 1976 and his M.S. and Ph.D. degrees in computer science from Carnegie-Mellon University in 1978 and 1983, respectively. His research centers on integrated intelligent systems (in particular, Soar), but also covers other areas such as machine learning, production systems, planning, and cognitive modeling.

Milind Tambe is a computer scientist at the University of Southern California Information Sciences Institute (USC-ISI) and a research assistant professor in the computer science department at USC. He completed his undergraduate education in computer science at the Birla Institute of Technology and Science, Pilani, India in 1986. He received his Ph.D. in 1991 from the School of Computer Science at Carnegie-Mellon University, where he continued his research as an associate until 1993. His research interests are in the areas of integrated AI systems, multi-agent systems, teamwork and agent modeling.