Flexible Group Behavior:
Virtual Commanders for Synthetic Battlespaces

Randall Hill, Jonathan Gratch, Paul Rosenbloom

University of Southern California

4676 Admiralty Way, Marina del Rey, CA 90292-6695

+1-310-822-1511

{hill,gratch,rosenbloom}@isi.edu

ABSTRACT

This paper describes a project to develop autonomous commander agents for synthetic battlespaces. The commander agents plan missions, monitor their execution, and replan when necessary. To reason about the social aspects of group behavior, the commanders take various social stances that enable them to collaborate with friends, exercise or defer to authority, and thwart their foes. The purpose of this paper is to describe these capabilities and how they came to be through a series of lessons learned while developing autonomous agents for this domain.

Keywords

Continuous planning, collaboration, social reasoning, stances, agents, synthetic battlespaces.

1. INTRODUCTION

Over the last four years we have been involved in a large effort to incorporate intelligent agent technology into state-of-the-art military simulation systems. The goal throughout this time has remained the same: develop command and control agents that can model the capabilities of a human military command. Our techniques and understanding of the issues, however, have evolved considerably through our experience with trying to make an organization of agents behave coherently in a complex, real-time, and definitely not benign environment. Natural Selection works in virtual worlds as well, and this paper describes the lessons we learned and how our techniques and preconceptions were shaped by the constraints of building a working system

2. BACKGROUND

Synthetic battlespaces provide a means of simulating combat situations at the level of the individual entity. An individual entity can be a truck (such as a supply truck), a tank, an aircraft (e.g., fighter jet, helicopter), an individual combatant (IC), or any other distinct, active object. When entities are combined with a synthetic environment – containing a terrain model, weather, time-of-day, illumination, dust, and smoke – and a distributed computing environment, advanced distributed simulation (ADS) becomes possible. The Department of Defense (DoD) is interested in using these advanced distributed simulations for training, where students are placed in simulators and participate in training scenarios; analysis purposes, providing a way of testing new doctrine and tactics; and mission rehearsal, where soldiers and commanders can run through a variety of scenarios in a simulator prior to running a real-world mission.

One of the biggest obstacles to creating high quality synthetic battlespaces is providing the requisite number and variety of forces with the degree of fidelity needed. It is relatively straightforward to model an autonomous entity such as a missile in flight using algorithmic methods, but it is much more complex to model entities that are supposed to exhibit realistic human behavior. Some behavior can be generated by instrumenting actual vehicles or inserting humans to drive avatars, but these solutions are prohibitively expensive in scenarios involving thousands of entities. The DoD has been interested in automating entity behavior in these simulations; such entities are alternately called *Synthetic Forces* or *Computer Generated Forces*.

Researchers have investigated varying levels of sophistication in their agent models. Semi-automated forces (SAF) provide relatively simple computer-generated entity behaviors, and then depend on human operators to provide higher level guidance and rescue when they get in trouble [2]. Intelligent Forces (IFORs) attempt to provide intelligent autonomous entity behavior that is broad, appropriate, and robust enough for use without human controllers [21]. Command Forces (CFORs) attempt to automate the commanders that sit above the entity level, thus providing automated models of command decision makers and automated tasking for entities [20].

Our own efforts over the past several years have been focused on research, development, deployment, and testing of "air" IFORs (fighters and helicopters) and CFORs (helicopter commanders) for the synthetic battlespace[1]. This work has been based on the Soar architecture [14], which held promise because of its

---

[1] Initially, researchers from USC-ISI and the University of Michigan jointly investigated IFORs for air-to-air fixed wing aircraft (i.e., fighters).

emphasis on the integration of intelligent capabilities and its dual status as both an architecture for constructing artificially intelligent systems and a unified theory of cognition [17].

3. LESSONS LEARNED

Our project built on earlier work in using IFOR models of fighter pilots and transferred this technology to the domain of helicopter agents. Initial work focused on pilot agents and has gradually evolved to modeling higher-level units in the command structure. Each step up uncovered a host of issues, only some of which we had anticipated.

The IFOR fighter pilots [12][21] and IFOR helicopter pilots [10][11] are capable of complex, autonomous behaviors in the synthetic battlespace. They carry out missions requiring them to interact with their own aircraft, other agents and the environment for hours at a time, with little or no interaction with human controllers. Bigger brains (moving from finite state machines to a reactive planning system) allowed IFOR fighter pilots to easily outperform SAFs in simulation. Reactive planners can represent more context, making it easier to ensure coherent action selection, especially when the situation departs from expectations. However, when we applied these models in the helicopter domain, a number of complications arose. Helicopter missions involve a great deal more coordination than fighter missions. Early application of the fighter model resulted in a number of highly visible breakdowns in coordination.

A simple domain choice, modeling helicopters, led to a substantial shift in research focus, namely understanding teamwork. The helicopter pilots were given the ability to act as a team [22]. This enabled them to respond to situations where teammates were lost or team goals were perceived to be unachievable, and it greatly enhanced the robustness of the agents' ability to continue to coordinate their actions when certain kinds of failures in the mission occurred. These IFOR teams addressed another main limitation of SAF models, which also exhibit coordination problems.

Although teamwork models resolved a number of limitations with regard to coordination issues, as we evaluated the situation in increasingly unscripted demonstrations, further limitations began to highlight the need for command and control capabilities. IFORs execute their mission based on a set of orders they receive. In many cases, the situation would change in ways that made these orders no longer appropriate. In many cases, the reasons for the failure were difficult to represent in the IFORs' reactive architecture as they fell more in the domain of deliberate planning. For example they might receive some information that would invalidate their future actions, but since reactive planners don't explicitly represent future activities or the causal relationships between actions, it is difficult to provide a general mechanism to recognize such failures.

Our initial version of a command and control agent, CFOR-Soar, focused on plan generation, but we soon found out that what is needed is a capability to plan continuously. In other words, just generating a plan was not sufficient – once the plan was disseminated, the commander had to monitor its execution as well, to insure that if a plan failed it would be able to make repairs. This also meant that the commander had to have a current world model that would provide the means of evaluating how well the plan execution was proceeding, and enable it to reason about

future failures. Hence, what was needed was the ability to interleave understanding (world model), planning, execution, plan monitoring, and replanning.

But even continuous planning turned out to be insufficient to produce flexible group behavior. While it did provide flexible, responsive decision-making, it lacked the kind of social awareness that any good human commander possesses. For example, in a scenario where the synthetic commander of one helicopter company saw that its sister unit was blindly headed for a previously undetected enemy unit, the commander did nothing to warn the other unit of the impending danger. This would obviously be unacceptable in a real-life situation. What was lacking was an ability to reason about the plans and goals of another group – the commander simply did not know how to recognize that its peer's company was threatened, nor would it have known what to do even if it did recognize the threat. It did not have the ability to reason at a social level. It became obvious that collaboration and other activities involving more than one agent requires an ability to reason about other agents and groups of agents.

Coherent, flexible group behavior does not automatically arise by simply collecting together a set of flexible individuals, even if they are working from the same mission plan – it requires both the ability to respond to contingencies, and the ability to reason about an entire group's status and the state of the world as it affects the group. In general, what is needed is the ability to understand and reason about other entities in one's own group as well as about other groups of entities, and the basis for interacting with others will vary in accordance with each one's role, the structure of the organization, and the way that decisions are made in the group.

In the remainder of this paper we will first give an example from the domain to illustrate the kind of planning and execution that is required. Next, in section 5 we describe continuous planning, which gives an overview of the architecture of the CFOR-Soar agent architecture. In section 6 we describe how the continuous planning capability was extended to enable CFOR-Soar to reason about the plans of other agents. Finally, we give an assessment of how well these techniques have worked out and where we're headed for the next phase of research.

4. DOMAIN EXAMPLE

We have implemented the commanders of an Attack Helicopter (Apache) Battalion. This includes the battalion commander (this agent simulates the commander plus other functions of the staff), and the Company Commanders of three Apache companies and a Combat Service Support (CSS) company, which performs logistical support functions for the battalion. We continued to use the IFOR helicopter pilots that had previously been developed to run the missions. Each Apache company consists of five to eight helicopters, while the CSS company is composed of a collection of ammunition and supply vehicles. The missions are planned and executed in JointSAF, a synthetic battlespace, by IFOR helicopter pilot agents and semi-autonomous force (SAF) vehicles (in the logistics and artillery roles.) The scenarios are typically run against battalion (30-50 tanks, armored vehicles, and trucks) and regiment-level (100-300 entities) groups of opposing force entities.

In a deep attack mission, the Apache battalion commander generates a battle plan, which is then sent as a standard formatted

military message called an Operations Order (OPORDER) to each of the company commanders (3 helicopter commanders and 1 logistics commander). The company commanders analyze the OPORDER and generate plans for their respective companies to cover their battle and logistics missions. Once the company level planning is complete, the company commanders back-brief their plans to the battalion commander, who analyzes them for conflicts. Once approval has been given by the battalion commander, the company commanders send the orders to their subordinates (pilots), who execute the mission.
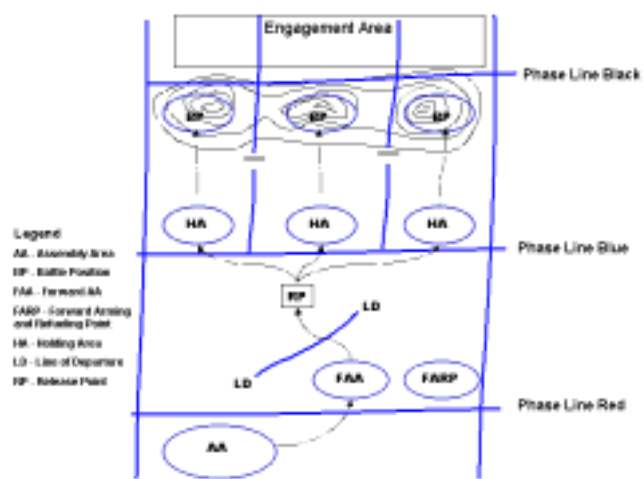
Figure 1: Example of Deep Attack scenario

A typical battalion-level deep attack mission is illustrated in Figure 1. The battalion begins the mission in an Assembly Area (AA). At a pre-determined time, the helicopter crews fly to a Forward Arming and Refueling Point (FARP), where the logistics vehicles from the CSS company re-supply the helicopters with ammunition and fuel. From there the crews move to the Forward Assembly Area until it is time to start the mission. The Battalion commander sends out a reconnaissance scout to observe the engagement area. When the scout reports the enemy forces passing a predetermined trigger line, the Battalion Commander initiates the deep attack – the companies take off and fly in formation along designated routes behind friendly lines until they reach a line of departure (LD). From this point they move through a reference point (RP), and then fly an assigned route to a holding area (HA), varying their formations, altitude, and speed in accordance with the battle plan. At the holding area, the majority of each company's helicopters wait while the scouts fly forward to perform a reconnaissance of the battle position (BP) – the scouts make sure it is safe for the company to occupy this preplanned area from which they will launch an attack, and they look for enemy units near the expected Engagement Area (EA). The BP shown in Figure 1 is located along a line of hills, which provide concealment for the helicopters from the EA. Once the scouts determine it is safe in the battle position and they observe enemy units passing into engagement area, they call forward the company. At this point the individual helicopters fly forward to their firing positions. The company commences a coordinated pop-up attack, where the helicopters unmask from their concealed positions, locate and lock onto a target with a laser designator, and fire a missile. After the missile has detonated, the helicopter crew can either choose to fire a second round at the same or another target, or they mask again and move to a slightly different

location. These pop-up attacks continue until the engagement criteria have been met – i.e., a certain percentage of the enemy vehicles have been disabled, the company is low on fuel or ammunition, or friendly helicopters have been damaged or destroyed. When one of these conditions occurs, the company returns to base, following a preplanned egress route.

Each of the Apache companies in the battalion performs the same type of mission, while the battalion commander monitors the progress of the execution. Throughout the mission the company commanders send reports to the battalion commander, who uses this information and other intelligence to keep track of the situation. In cases where the battle plans have to be modified because of a contingency (e.g., the enemy isn't located where they were expected to be), then the commanders may have to replan parts of the mission. For this reason it is important for the commanders to understand the situation – where the enemy units have been spotted, what they are doing, where their own subordinates are currently located, how far they have progressed in the execution of their plans, and whether there are conditions that prohibit the plans from being completed successfully. There are many different factors that could make it necessary to replan.

While the core tasks of the deep attack mission are performed by the helicopter companies, the battalion commander also plans the activities of the combat service (CSS) support company, which provides the ammunition and fuel at the FARP. In addition, the battalion commander has to coordinate with other units, including requesting the artillery to perform a mission called Suppression of Enemy Air Defense (SEAD). SEAD support is planned against known enemy air defense unit locations along the route to be taken by helicopter companies – the purpose of the mission is to keep the air defense units from engaging the helicopters when they fly through the vicinity.

5. CONTINUOUS PLANNING

Modeling the behavior of commanders and their staffs (collectively these decision-making and coordination functions are called Command and Control or C2) requires a *continuous planning* capability. C2 agents must perform a number of functions typically associated with planning algorithms. They must develop sequences of tasks in the service of mission goals, they must understand the constraints and assumptions underlying these plans, and they must reason about potential interactions between their plans and those of other entities. C2 agents must also perform a number of functions more commonly associated with reactive systems. They must continuously assess the situation and react to unexpected events. To support this mixture of requirements, we developed a planning system (CFOR-Soar) based on a set of planning techniques that integrated planning, execution, and repair [1][5][6][15][23].

CFOR-Soar plans hierarchically. The *domain theory* contains knowledge about tasks – preconditions, effects, interruption conditions, the probability of success, the utility (importance), responsible entity (who performs the task), and a sequence of procedures that should fire during the execution of the task. In addition, the domain model contains decomposition schemata that describe how to decompose a task into simpler tasks. For example, a deep attack mission can be decomposed into tasks to move to the holding area, move to and occupy the battle position, engage the enemy, and return to base. Each of these tasks may be
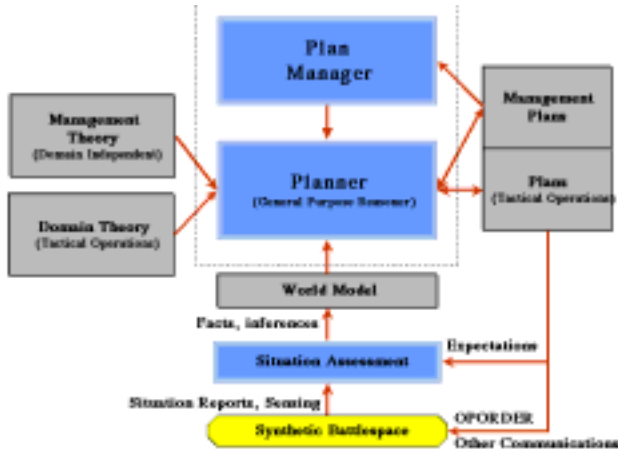
Figure 2: CFOR-Soar Commander Architecture

further decomposed – e.g., the task for moving to the holding area may consist of several smaller move tasks.

During the *plan generation phase*, the CFOR-Soar *planner* receives an Operations Order, which is itself a partial plan specification that provides high-level guidance on how to complete the mission. The planner must refine this specification into a concrete battle plan for the forces in the planner's unit. The planner varies the way it decomposes tasks depending on the current or projected situation – the subtasks that are chosen may vary and can be thought of as alternative courses of action to accomplish the high level task. Once the planner chooses how to decompose a task, the context validating this decision is recorded in the plan so that if the context later changes, the planner can later verify whether the decomposition is still appropriate. For example, a helicopter company commander may choose to fly in a column (maximizing speed) because it believes there are no enemy forces on its avenue of approach. If subsequent intelligence contradicts this assumption, the CFOR-Soar planner can recognize that a slower but safer formation is more appropriate.

During the *plan execution phase*, CFOR-Soar builds a *world model*, which represents its interpretation of the current situation. The world model is built and maintained by using domain-specific routines to perform what is commonly known as *situation assessment* on information from the synthetic battlespace. The battlefield information is provided by on-board sensors and situation reports. The planner continuously compares the world model against its current plans, and uses these comparisons as inputs to the reasoning that underlies *plan monitoring* and *repair*. For example, if a CFOR-Soar battalion commander receives a report that one of its companies has reached the holding area, the planner recognizes that this information satisfies the completion condition of the ingress task. This in turn allows the planner to infer that the ingress has terminated and the company is now prepared to engage. In contrast, a report that the company is delayed might violate the current constraints in the plan and force some repair activities. So if the delayed flight was to engage in a coordinated attack with another company, the battalion commander might delay the second company's departure, or even cancel the entire mission.

During plan execution, the planner only initiates tasks whose preconditions are satisfied in the world model (and are not preceded by any uninitiated tasks). Similarly, the planner terminates tasks when all of their effects appear in the world. Task initiation and termination may be *interleaved* with other planning operations. As the world model reflects the perceived state of the world, it may change in ways not predicted by the current plan network. For example, if the battlefield environment changes in unpredicted ways, these changes may provide opportunities (as when an unsatisfied precondition is unexpectedly observed in the world). They may also threaten constraints in the task network, forcing the planner to *repair* the plan to resolve them.

Figure 3 illustrates a set of tasks maintained in a task network. Each task has a set of preconditions (predicates listed at the bottom left of each task) and a set of effects (predicates listed at the bottom right of each task). A valid plan must ensure that each precondition is established by some effect. In the figure, horizontal bars correspond to the protection constraints. Each protection constraint represents the fact that an effect is being used to establish a precondition, and that the effect must be protected throughout the duration of the protection constraint. In this example, the helicopter company's plan is a sequence of two tasks: (1) company A moves to the battle position, Move(A,BP), and (2) company A engages the enemy, Engage(A,Y), where Y represents the enemy. The preconditions for moving to the battle position are: (1) that company A must be located in the forward assembly area, at(A,FAA), and (2) there must be fuel at the forward assembly area, at(fuel,FAA). These preconditions must both be true in order for this task to be performed – if either of them does not hold then the task cannot be successfully completed.
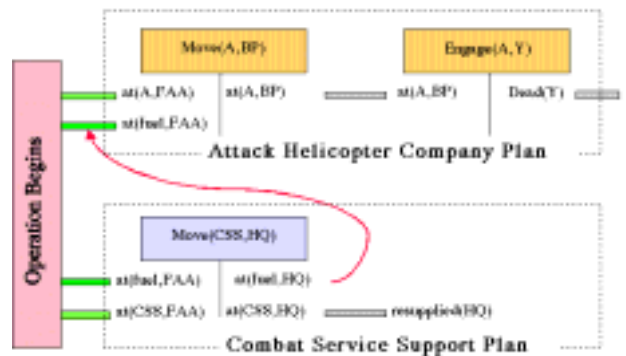
Figure 3: Example of Plans and Interactions

CFOR-Soar repairs plans using general repair operators that allow it to non-chronologically retract problematic constraints from the task network. The CFOR-Soar planner augments this capability by incorporating a validation-structure approach to plan repair [13]. The planner has a number of operations that allow it to modify its current plans. Some of these operations add constraints to the plan network, while other operations retract constraints.

6. COLLABORATION

Mission planning and execution is a collaborative enterprise involving agents distributed across multiple levels of an organization. Not only must a commander agent be capable of continuous planning, but it must also model the goals and plans of others, and reason about how its decisions will affect them. Factors such as the agent's role and the management culture of the organization will affect the interactions among entities, and it will ultimately change the overall behavior of the organization. Without collaboration, autonomous group behavior is not possible. Like continuous planning, collaboration requires an understanding of the tactical domain, to include models of domain-specific communications, decision-making protocols, organizations, and relationships. Given these models, it should be possible to represent a range of different organizations.

CFOR-Soar's collaborative capabilities build on a number of AI techniques. Agents exchange partial plans in a manner similar to Decker and Lesser's GPGP [4]. Pollack [19], and Grosz and Kraus[9] have described how multi-agent planning involves a level of meta-reasoning to manage the coordination of different agents. We've adopted a similar approach but specialized it to reason about the particular organizational constraints involved in military missions . We have also extended these models to account for the less than collaborative interactions that can occur between agents in military situations.

Collaboration can occur when a group has a common interest or goal that the members are willing to pursue together, even when it means modifying one's own goals and plans for the sake of the group. Not all social interactions are collaborative, however. For example, engaging an enemy force is an adversarial activity intended to *thwart* another's goals. In between collaboration and adversity there are many other forms of interaction that range from indifference to rudeness, which have an impact on the outcome of a social situation, whether it is simulated or real. Therefore, to model a broad range of group behaviors goes beyond representing collaboration and extends to other social interactions as well.

To achieve collaboration and other forms of social reasoning, the basic architecture of the planner had to be extended in several ways. First, one of the basic requirements for collaboration is that the agent must reason about the plans of other agents, thus it was extended to maintain *multiple plans in memory* and reason about their interactions. Second, to act as a member of an organization requires understanding how decisions are made and executed by the organization as a whole. This capability has been implemented by *explicitly representing the decision-making process* so that it can be taken into account when deciding how to interact with others. Third, the agent's role in an organization or social setting will affect the way it reasons, behaves and interacts with others. We have defined a set of *social stances* that affect the planner treats others when it makes its own plans. Finally, a *plan manager* was added to the planner as a way of integrating these features. Each of these extensions is discussed in greater detail in the paragraphs that follow.

6.1 Multiple Plans in Memory

First, the planner *maintains multiple plans in memory and reasons about their interactions*. This allows a command agent to not only reason about its own activities, but also represent, to some level of detail, the activities of friendly units and the projected activities of the opposing force. It gives the commander agent a more coherent picture of the overall situation and allows the agent to understand the relationships between plans and the effect on other units if a plan is changed.

Returning to the example shown in Figure 3, note that the planner is actually reasoning about two plans – the first is the plan of the Attack Helicopter Company and the second is that of the Combat Service Support Company. The planner detects a potential interaction between the two plans – the conflict is noted by the arrow pointing from one plan to the other. The Combat Service Support Plan has one task, to move the fuel currently located in the forward assembly area to the headquarters area. This plan directly conflicts with the Helicopter Company's plan, since it would move the fuel needed by the helicopters before they have a chance to refuel, thereby denying them the ability to move to the battle position. By maintaining multiple plans in memory and comparing their preconditions and effects it becomes possible to detect interactions and conflicts among the plans, which is the first step toward socially motivated planning.

6.2 Represent Decision-Making Process

The planner *maintains explicit representations of the decision-making process and the related organizational interactions*. These activities provide structure to the planning process and implement protocols for how and when distributed planning agents should exchange information. For example, the Army has formalized the way planning is done in what is known as the Military Decision Making Process (MDMP). MDMP breaks planning into a sequence of phases: mission analysis, course of action development, course of action analysis, course of action comparison, course of action approval, and orders production. Once the orders have been distributed, the execution phase begins, which may involve further planning (as described in the section on continuous planning).
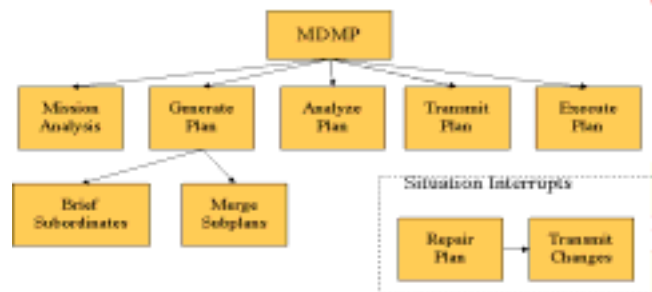

Figure 4: Military Decision Making Process (MDMP)

We explicitly represent each of the MDMP phases as tasks – they differ from those tasks usually considered by traditional planning systems as they refer to stages of the planning process, rather than primitive tasks an agent performs in the world. In so doing, we have implemented a form of *plan management* [16], which is typically viewed as meta-reasoning about plans and has been traditionally either ignored or modeled with very different algorithms and data structures than those used in planning. In the CFOR-Soar planner, these plan management activities are

represented as an explicit plan and are modeled using the same data structures as other domain activities. The inputs and outputs of these plan management tasks, in turn, determine the flow of information among agents in the organization. The advantage of this scheme is that (1) interactions among planning agents can be programmed as easily as other domain activities, (2) they can be programmed using the same data structures, and (3) they provide a uniform medium for supporting visualization and traceability of the reasoning process.

6.3 Social Stances

Third, the planner *models the social stances that the agent can take toward others at different stages of the planning process*. Specifically, a domain modeler can vary the degree to which a planning agent will be cooperative or antagonistic to the activities of other agents.   Factors that influence how the members of a group act toward one another and how a group as a whole behaves includes: the member's role (e.g., leader, follower, peer), organizational structure (e.g., hierarchical, matrix, flat, network), and management/group culture (i.e., how decisions are made and carried out). We hypothesize that each of these factors can be modeled through the use of *social stances*. A social stance is an inter-agent posture that affects how the agent reasons about and interacts with other individuals or groups.

- *adversarial:* models the posture taken toward one's enemies to thwart their intentions and plans by denying them access to a perceived goal.

- *authoritarian:* leaders in a hierarchical organization may take this stance to change to the goals and plans of a subordinate without negotiation or permission.

- *deferential:* subordinates take this stance toward a superior in the chain of command – when the boss says to change a plan, it is changed without any negotiation.

- *helpful/fair:* models the relationship with a friend or peer where one seeks to find resolutions to conflicts detected in their plan.

- *rude:* represents a planner that knows about a conflict in another's plan but selfishly hoards a resource that would have helped them, even when a compromise was possible.

- *blind:* ignores conflicts and interactions among selected plans.

6.4 Plan Manager

These three characteristics—maintaining multiple plans in memory and reasoning about their interactions, maintaining explicit representations of the decision-making process and the related organizational interactions, and modeling the social stances that the agent can take toward others at different stages of the planning process—are supported by a *plan manager* that augments the planner's basic reasoning capabilities. The plan manager keeps track of the fact that different tasks in the plan network correspond to the activities of different agents. Tasks are organized into a higher-level data structure called "a plan." Plans are intended to refer to clusters of activities that are meaningful in a particular domain. In a multi-agent application, different plans most naturally refer to the planner's understanding of the activities of different agents (e.g., my plans vs. my enemy's plans). The plan manager reasons about interactions between

plans and can alter the way the planner behaves towards different plans in the plan network.

Social stances are implemented by (1) determining which plan interactions are attended to, and (2) constraining the way the planner may modify different plans in the plan network via a *planning stance*.   By default, the planner attempts to resolve every perceived interaction in every plan it represents. The social stance one takes toward another agent, however, determines whether or not to pay attention to the interactions – some interactions may be filtered out – and how to deal with them should they occur. If the social stance calls for the planner to pay attention to a particular interaction, then the planner decides whether or not it can *modify* or *execute* the plan. These two planning stances are control properties that indicate how the planner treats the plans in its plan network.

The simplest example is the *blind* social stance – the planner does not pay attention to plan interactions at all, so there is no constraint placed on changing one's own plans since interactions with others' plans is not taken into consideration.

The deferential stance is used in modeling the military management culture – one has to accept orders from a commander. These orders must be obeyed, but one has some flexibility in fleshing out the details. A subordinate planning agent should distinguish between the part of the plan that is fixed and the part that it has the authority to alter, if, for example, the plans must be repaired during the execution phase. This can be modeled by representing overlapping plans. One plan contains the initial orders and is deemed unmodifiable but executable through a suitable choice of plan properties.   This plan is contained within a larger plan that allows modifications. Any changes made by the subordinate agent only appear in the larger plan, and the initial orders must remain unchanged.

Up to this point we've only discussed how to represent different social stances. However the agent also requires the ability to change stances *dynamically* as plans are generated and executed. For example, to implement the military decision-making process, an agent must take a modifying stance towards the mission plan until it has evolved to a satisfactory level. At that point, it must commit to these plans (taking an unmodifiable planning stance), share them with the troops, and make them available for execution (taking an executing stance).

If plans break down, the commander must return to a modifying stance until the plan is repaired. For instance, changes in the environment can invalidate current plans and replanning occurs in a layered fashion. Plans become more specific as one moves down the chain of command. This means a subordinate has some latitude in executing and repairing a plan while staying within the constraints mandated by their superiors.   This latitude is implemented by the appropriate definitions of plan management tasks. If a plan failure exceeds the scope of this authority (as when they require modifying the partial plan given to the subordinate), the unit's commander must detect the flaw, repair the plan, and communicate the change to its subordinates.

Dynamic stances are modeled by allowing plan properties to be mentioned and modified by tasks in the plan network. In this way we can create explicit *plan management plans* that are generated and executed just as any other plan handled by the planner. The only difference is that the preconditions and effects of such plan management plans refer to properties maintained by the plan

manager and their execution signals the plan manager to alter the current set of plan properties.

Through the use of social and planning stances, the CFOR-Soar can model an organization of agents that plans in a distributed and asynchronous manner. Different organizational structures are easily represented as input to the planner: one can manipulate the number and type of elements, how they exchange information, and the authority relationships between them. The architecture also supports differing levels of autonomy between commanders and their subordinates, thereby facilitating the modeling or more or less rigid organizational structures. For example, current military doctrine specifies a relatively rigid and hierarchical distributed planning process. This doctrine is represented in Soar/CFOR as a data structure, rather than being reflected in the planning architecture, making it is relatively easy to program in alternative organizational structures.

Each commander represents several plans in a single task network: there are base-level plans for each of the agents the commander knows about. For example, a company commander will have a base-level plan for its own activities, those of its sibling company, and those of any enemies it has been informed of. Each commander also maintains a plan management plan that explicitly implements the military decision making process.

7. EVALUATION

The helicopter pilots and an early version of the CFOR-Soar commander was demonstrated in an exercise known as the Synthetic Theater Of War (STOW-97) Advanced Concept Technology Demonstration. STOW-97 was a large, distributed, entity-level simulation. Five US sites participated – one each for the Army, Navy, Marines, Air Force (which also included Navy and Marine Air), and Opposing Forces – plus one in the United Kingdom. All told there were approximately five hundred computers networked together across these sites generating on the order of five thousand synthetic entities representing tanks, helicopters, airplanes, individual soldiers, surface ships, submarines, missile batteries, trucks, buses, etc.

The forty eight hours of the synthetic exercise included a range of missions and complex interactions among these various entities. Three Army deep attack missions were pre-planned, but at the last minute each of these missions had to be re-planned since the original plans were 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 destroyed, while three helicopters were officially lost.

Based on our STOW-97 experiences, we extended the CFOR-Soar agents to include the continuous planning and social stances described in this paper. In the course of developing these agents we have run many hundreds of scenarios. Over time we have refined the behaviors of the CFOR-Soar commander and RWA-Soar pilot agents so that they now handle many variations of the basic scenario illustrated in Figure 1.

From our experiences with CFOR-Soar, we have concluded that the continuous planning capability works well as a whole. In most of the scenarios we've run where the enemy forces are in approximately the position that was predicted, the deep attack missions were conducted successfully. The commander agents monitor and track the progress of their respective subordinates, and in some scenarios, when it became necessary, the commander repairs and resend the battle plans. For instance, in the case where a commander received a report about enemy air defense threatening the company's path ahead, CFOR-Soar successfully replanned the route and avoided harm. These kinds of cases are still somewhat limited in number, however, due to the amount of time it takes to acquire and encode the knowledge.

A weakness in CFOR-Soar's continuous planning capability is in the area of situation awareness. Perhaps it was because we did not spend enough time addressing this issue, or it may have been because situation awareness tends to be very domain specific. In either case, when the CFOR-Soar commander did not recognize what was going on in a situation, then it could not respond in a completely appropriate manner. For example, in one scenario, a small unit of enemy tanks was spotted by an advance scout near the company's battle position. The scout, however, did not spot the main body of the enemy forces located in the engagement area. The company commander assumed that the forces at the battle position was all that was to be attacked, hence, once it had successfully engaged them it went home without engaging the rest of the force. Both the scout and the company commander lacked the ability to reason about the situation so that they could infer than there must be more forces out there and that they should look for them and engage them. Situation awareness and understanding are critical for building a robust continuous planner.

In the area of collaboration, we found that the model of social reasoning worked well in cases dealing with well defined roles such as one finds in the stereotypical military unit. In a perfect world where commanders wield authority and subordinates defer to their superiors, the social stances are sufficient to model a lot of what typically goes on in a military unit with respect to collaboration. Furthermore we successfully modeled helpfulness in the scenario where a commander notices that his sibling company will soon be threatened by some enemy forces – as a result of social reasoning it contacted the commander agent with the vital information.

To improve the current model of collaboration we would like to extend the model to use the teamwork protocols proposed by Cohen and Levesque [3] and implemented by Tambe [22] in the helicopter pilot agents. On more than one occasion a helicopter company would wait forever for a report to arrive from a scout who happened to have crashed while performing reconnaissance. The same kind of teamwork protocol that was used in the pilot agents for deciding when to give up on a commitment would help the CFOR-Soar agent to know where it is time to find another way to accomplish a goal.

8. CONCLUSIONS

In the course of building commander agents for large scale military simulations, we have learned that to build organizations of agents capable of flexible, coherent group behavior requires the ability to plan continuously and to plan with social stances in mind. Continuous planning is a way of interleaving planning, execution, monitoring, understanding, and repair so that the agent not only deliberates about the future, but makes changes to its plans on the fly, based on being aware of threats to the plan in the current or a future situation. Planning with social stances enables

an agent to not only reason about the plans of others, but it enforces organizational and social relationships and their impact on the decision-making process.

We plan to extend these capabilities by adding other behavior moderators found in humans such as emotion and personality. In the end we hope to have good representations of human behavior, both as individuals and as groups.

9. ACKNOWLEDGEMENTS

10. REFERENCES

[1]	Ambros-Ingerson, J. A. and Steel, S. 1988. "Integrating Planning, Execution and Monitoring," in AAAI-88.

[2]	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.

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

[4]	Decker, K. S. Lesser, V. R., 1992. "Generalizing the partial global planning algorithm." *Int. Journal of Intelligent and Cooperative Information Systems, 1 (2).*

[5]	Fikes, R. E. and Nilsson, N. J., 1971. STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence 2 (3-4). 189-208.

[6]	Golden, K., Etzioni, O., and Weld, D. 1994. "Omnipotence without Omniscience: Efficient Sensor Management for Planning

[7]	J. Gratch. "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.

[8]	J. Gratch, 1998. "Reasoning about Multiple Plans in Dynamic Multi-Agent Environments," in Proc. of AAAI Fall Symposium on Distributed Continual Planning, Orlando, FL.

[9]	Grosz, B., and Kraus, S. 1996. "Collaborative Plans for Complex Group Action," Artificial Intelligence, 86(2).

[10]	R. Hill, J. Chen, J. Gratch, P. Rosenbloom, M. Tambe, "Soar-RWA: Planning, Teamwork, and Intelligent Behavior for Synthetic Rotary Wing Aircraft," *Proceedings of the 7th Conference on Computer Generated Forces & Behavioral Representation*, Orlando, FL., May 12-14, 1998.

[11]	R. Hill, J. Chen, J. Gratch, P. Rosenbloom, M. Tambe, "Intelligent Agents for the Synthetic Battlefield: A Company of Rotary Wing Aircraft," *Proceedings of Innovative Applications of Artificial Intelligence* (IAAI-97), Providence, RI, July 1997.

[12]	Jones, R. M., Laird, J. E., & Nielsen, P. E. (1998). Automated intelligent pilots for combat flight simulation. Proceedings of the Tenth Annual Conference on Innovative Applications of Artificial Intelligence (pp. 1047-1054). Menlo Park, CA: AAAI Press.

[13]	Kambhampati, S. and Hendler, J. 1992. A validation-structure-based theory of plan modification and reuse. *Artificial* Intelligence 55, pp. 193-258.

[14]	Laird, J., Newell, A., Rosenbloom, P. "Soar: An architecture for general intelligence," Artificial Intelligence 33(1):1-64, 1987.

[15]	McAllester, D. and Rosenblitt, D. 1991. "Systematic Nonlinear Planning," in AAAI-91.

[16]	Myers, K. L., 1998. Towards a Framework for Continuous Planning and Execution. AAAI Fall Symposium on Distributed Continual Planning, Orlando FL.

[17]	Newell, A. *Unified Theories of Cognition.* Harvard Press, 1990..

[18]	Mackworth, A. The logic of constraint satisfaction. *Artificial Intelligence* 58, 1992.

[19]	Pollack, M.E., 1992. "The uses of plans," *Artificial Intelligence, 57(1)*, pp 43-68.

[20]	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

[21]	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.

[22]	Tambe, M. 1997. Towards Flexible Teamwork. Journal of Artificial Intelligence Research, Volume 7, pp. 83-124.

[23]	D. Wilkins. Practical Planning. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1988.