Teamwork in Real-world, Dynamic Environments

Milind Tambe
Information Sciences Institute
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292
tambe@isi.edu

July 11, 1996

Topic AREAS: Cooperation and coordination, Pratical applications

Abstract

Flexible teamwork in real-world multi-agent domains is more than a union of agents' simultaneous execution of individual plans, even if such execution pre-coordinated. Indeed, uncertainties in complex, dynamic domains often obstruct pre-planned coordination, with a resultant breakdown in teamwork. The central hypothesis in this paper is that for durable teamwork, agents should be provided explicit *team plans* and an underlying model of teamwork that explicitly outlines their commitments and responsibilities as participants in team activities. Such a model enables team members to flexibly reason about coordination activities. The underlying model we have provided is based on the *joint intentions* framework; although we present some key modifications to reflect the practical constraints in (some) real-world domains. This framework has been implemented in the context of a real-world synthetic environment for helicopter-combat simulation; some empirical results are presented.[1]

[1]This research was supported as part of contract N66001-95-C-6013 from ARPA/ISO. Domain expertise was provided by Dave Sullivan of BMH Inc. I thank Ramesh Patil, Wei-Min Shen and Takahira Yamaguchi for helpful comments on an earlier draft of this paper.

1 Introduction

Many AI researchers are today striving to build agents for complex, dynamic multi-agent domains, such as, virtual theatre[7], realistic virtual training environments (e.g., for emergency drill[17] or combat[22, 19]), virtual interactive fiction[1], RoboCup robotic and virtual soccer[13] and robotic collaboration by observation[14].

Most of this research has so far focused on enabling *individual agents* to cope with the complexities of these dynamic domains. One promising approach that has emerged is the use of *hierarchical reactive plans*. Reactive plans are qualified by preconditions, which help select plans for execution based on the agent's current high-level goals/tasks and beliefs about its environment. Selecting high-level abstract plans for execution leads to subgoals and thus a hierarchical expansion of reactive-plans ensues. Activated plans terminate via terminating conditions. Agents built in architectures such as PRS[9], BB1[7], RAP[5] and Soar[16] for dynamic domains may be (at least abstractly) characterized in this fashion.

Instead of individuals, this paper focuses on *agent teams* in dynamic domains. All around in our daily lives, we participate, interact or observe dynamic team activities, such as, driving in a convoy, participating in team sports (e.g., soccer), enjoying plays (theatre) and discussions, or watching televised military exercises. These activities are being reflected in many of the multi-agent domains discussed above. Such team activities are not merely a union of simultaneous, coordinated individual activities[6, 3]. For instance, ordinary automobile traffic is not considered teamwork, despite the simultaneous activity, coordinated by traffic signs[3]. Indeed, our commonsense notion of teamwork involves more than simple coordination, e.g., the American Heritage Dictionary defines it as *cooperative effort by the members of a team to achieve a common goal.*

Yet, to sustain such cooperation in complex, dynamic domains — whether it is driving in a convoy or playing Soccer — agents must be flexible in their coordination and communication actions, or else risk a breakdown in teamwork. To achieve such flexibility we apply one key lesson from the arena of knowledge-based systems — an agent must be provided explicit "deep" or causal models of its domains of operation [4]. The key here is to recognize that when an agent participates in a team activity, teamwork is itself one of the domains, and hence the agent must be provided an explicit model of teamwork. Unfortunately, in implemented multi-agent systems, team activities and the underlying model of teamwork are often not represented explicitly[10, 11]. Instead, individual agents are often provided individual plans to achieve individual goals, with detailed precomputed plans for coordination and communication. However, in real-world dynamic environments, unanticipated events — such as an unexpected interruption in communication — often disrupt preplanned coordination, jeopardizing the team's joint effort (Section 2 provides detailed examples).

The recent formal theories of collaborative action have begun to provide the required models for flexible reasoning about team activities[3, 6, 12, 11]; although few multi-agent implementations have built up on them[11] (a notable exception is [11], described in Section 6). In contrast, this paper describes an implemented, real-world multi-agent system that builds upon one such model. Our central hypothesis is that for effective teamwork in complex, dynamic domains, individual team members should be provided *reactive team plans*, that explicitly express a team's joint activities — although these may hierarchically expand out into reactive plans for an individual's role in the team. To execute such team plans, team members must be provided an explicit model of teamwork — their commitments and responsibilities as team members — so they can flexibly reason about coordination and communication. In our work, this model is the formal *joint intentions* framework[3], which we have modified in key ways to accommodate the constraints that appear typical in (some) real-world dynamic domains.

Before describing reactive team plans in detail, we first concretely motivate their need by describing our initial experiences in designing agent teams for a real-world domain. Given our focus on a real-world multi-agent domain — with key characteristics such as dynamism and realistic communication costs that are representative of other real-world domains — the lessons learned here appear to have wider significance. All our implementations are based on the Soar architecture[16, 20]. We assume some familiarity with Soar's problem-solving model, which involves applying an operator hierarchy to states to reach a desired state.

2 A Real-world Domain and Initial Experiences

We are building intelligent pilot agents for synthetic aircraft in a battlefield simulator, commercially developed for the military for training[2]. These pilot agents have participated in large scale combat exercises, some involving expert human pilots[22]. This paper will focus on pilot agents for a company of (up to eight) attack helicopters, which execute missions in a synthetic 3D terrain with hills, valleys and ridges (e.g., southern California) [24].[2]

As shown in Figure 1, in a typical attack mission, the company may fly 25-50 kilometers at varying altitudes, to halt at a holding point. One or two *scout* helicopters in the company fly forward to check the battle position, i.e., the location from where the company will attack enemy forces. Once the battle position is scouted, other members of the company move forward, each hovering in its own designated subarea of the battle position. Here, an individual pilot agent hides/masks its helicopter. To attack, the pilot has his helicopter "popup" (rise high), to shoot missiles at enemy targets. The helicopter then quickly masks and moves as protection against return fire, before popping up again. When the mission completes, the helicopters regroup and return to base.

In our first implementation of the helicopter company, each pilot agent was provided an operator (reactive plan) hierarchy to execute its mission[24]. Figure 2 illustrates a portion of this operator hierarchy (at any one time, only one path in this hierarchy from the root to a leaf node is active). Each operator consists of (i) precondition rules, to help select the operator; (ii) application rules to apply the operator once selected (a high-level, non-leaf operator may subgoal); (ii) termination rules, to terminate the operator.

To coordinate among multiple pilot agents we used techniques quite comparable to previous such efforts, including our own, in the synthetic battlefield domain[22, 18, 25]. In particular, each

[2]This basic simulation technology, once proven promising in *training* for military applications, has led to other possible applications ranging from training for disaster relief to interactive entertainment.

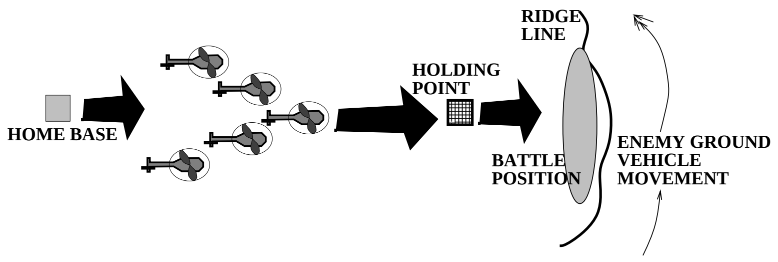

Figure 1: A company of helicopters in simulated combat. The ridge line is ideal for masking.

individual was provided specific plans to coordinate with others. For instance, when at the holding point, the scout first executed an operator to fly to the battle position, and then another operator to inform those waiting at the holding point that the battle position is scouted. Similarly, to fly in formation, each agent was assigned a "partner" agent to follow in formation (unless the agent was leading the formation). Eventually, all coordination within a group was accomplished by each agent coordinating with its partner.

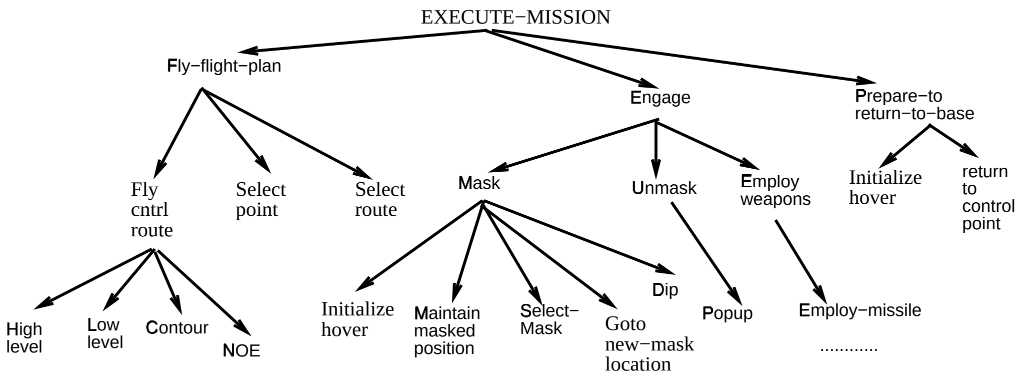

Figure 2: A portion of the operator hierarchy for an individual helicopter pilot agent.

The resulting pilot agents each contained about 1000 rules, and the company was tested in October 1995 in a three-day exercise (with upto 400 agents in the synthetic battlefield). While the helicopter company executed helicopter tactics adequately, the exercise revealed some key problems in teamwork — see Figure 3 for some illustrative examples.[3]

While a programmer could add specialized coordination actions to address the above failures once discovered, anticipating such failures is extremely difficult, particularly as we scale-up to increasingly complex team missions. Instead, the approach pursued in this work is to focus on the root of such teamwork failures — that as with other multi-agent systems, individual team members have been provided fixed coordination plans, which break down when unanticipated events occur. In particular, the team goals and/or team plans are not represented explicitly. Furthermore, an underlying model of teamwork, spelling out team members's commitments and responsibilities

[3]This demonstration was done jointly with Paul Rosenbloom and Karl Schwamb.

1. Upon reaching the holding area, the company waited, while the scout started flying forward. Unfortunately, the scout unexpectedly crashed into a hillside. Hence, the rest of the company just waited indefinitely at the holding area, waiting to receive a message from the (crashed) scout that the battle position was scouted.

2. Upon recognizing that the mission was completed, one company member (the commander) returned to home base, abandoning others at the battle position. The commander's "partner" agent was unexpectedly shot down, and hence it failed to coordinated with others in its company.

3. While attacking the targets from the battle position, only one member of the company could see the targets. Thus, only one member engaged the targets; the others returned without firing a single shot.

4. Some company members failed to recognize that they had reached a waypoint — the agent leading the formation had reached the waypoint, but those trailing in formation concluded they had not individually done so (despite tolerance ranges in measuring distances).

Figure 3: Some illustrative examples of breakdown in teamwork.

towards others when executing a team activity, is absent. That is why, for instance, an agent ends up abandoning its team members in a risky situation (Item 2, Figure 3). That is also why the company cannot recover when the scout crashes (Item 1, Figure 3) — there is no explicit representation of the company's team goal at the holding point and the scout's part in it.

3 Explicit Model of Teamwork

To provide agents with an explicit model of teamwork, we rely on the *joint intentions* framework[3, 15], since currently it is perhaps the most well-understood framework. In this framework, a team Θ jointly intends a team action if team members are jointly committed to completing that team action, while mutually believing that they were doing it. A joint commitment in turn is defined as a joint persistent goal (JPG). A JPG to achieve **p**, where **p** stands for completion of a team action, is denoted JPG(Θ, **p**). JPG(Θ, **p**) holds iff three conditions are satisfied[4]:

1. All teammembers mutually believe that **p** is currently false.

2. All teammembers mutually know that they want **p** to be eventually true.

3. All teammembers mutually believe that until **p** is mutually known to be achieved, unachievable or irrelevant, they mutually believe that they each hold **p** as a weak goal (WG). WG(μ, **p**, Θ), where μ is a team member in Θ, implies that μ either (i) Believes **p** is currently false and wants it be eventually true (i.e., **p** is a *normal achievement goal*); or (ii) Having privately discovered **p** to be achieved, unachievable or irrelevant, μ has committed to having this private belief become Θ's mutual belief.

[4]JPG(Θ, **p**) also includes a common escape clause q, omitted here for the sake of brevity.

Two important issues should be noted. First, there is a change in expressiveness of plans — in this framework, an entire team can be treated as jointly committing to a team plan. For example, when a company of helicopters flies to a waypoint, it is a team jointly committing to a team activity — each individual is not flying on its own to that waypoint, while merely coordinating with others. Thus, it is sufficient if the team reaches the waypoint, each individual need not do so individually[5]. Such a change in plan expressiveness alleviates concerns such as the fourth item in Figure 3.

Second, to establish a joint intention, agents must hold a WG (weak goal) which ensures that members cannot freely disengage from their joint commitment at will. In particular, while a JPG(Θ,**p**) is dissolved when a team member μ privately believes that **p** is either achieved, unachievable or irrelevant, μ is left with a commitment to have this belief become mutual belief. To establish mutual belief, an agent must communicate with other team members. While this communication is an overhead of team activity, it enables an individual to ensure that its teammates will not waste their time or face risks unnecessarily. This alleviates difficulties such as the second example in Figure 3, where an individual disengaged from the joint commitment without informing other team members, and exposed them to unnecessary risks.

This framework provides an underlying model of teamwork, enabling flexible reasoning about coordination activities. For instance, there is an explicit justification for communication, enabling agents to reason about it. The following now presents some key modifications to accomodate real-world constraints. Even though we draw upon examples from our domain, we expect similar issues to arise in other dynamic environments. (Operationalization of these ideas described in Section 4).

3.1 Modifying Commitments

Fulfilling the requirements in WG(μ,**p**,Θ) requires a team member to unconditionally commit to communicating with other team members, whenever it drops **p** as a normal achievement goal. However, in many environments, such as synthetic battlefields or soccer fields, communication can be costly, risky or otherwise problematic. For instance, in battlefield simulations, communication may break radio silence, severely jeopardizing a team's overall joint activities. Therefore, the unconditional commitment to communication is modified to be conditional on communication benefits to the team outweighing costs (to the team). Also included in this modification is an agent's commitment to search for alternative lower-cost methods of communication (e.g., the agent may travel to personally deliver the message, if using the radio is risky). Nonetheless, in some cases, benefits will be outweighed by costs, and hence no commitment to communication will result. In other extreme cases, an agent may be simply disabled from communication even after dropping its normal achievement goal (e.g., a pilot may be shot down).

Such communication difficulties require that other team members take up some of the responsibility for attaining mutual belief. In particular, a team member must attempt to track the team's beliefs in the status of their joint goal. For instance, if a company of helicopters reaches a well

[5]This may mean that the first or some pre-specified percentage of vehicles reach close to the waypoint.

specified waypoint, the team can be tracked as recognizing its achievement, and thus unnecessary message broadcasts can be avoided.

A second modification focuses on the dissolution of a joint commitment (JPG). In particular, currently, if an individual μ is known to drop the normal achievment goal, the joint commitment is automatically dissolved. Yet, such an automatic dissolution is often inappropriate. For instance, if one helicopter μ in the company of eight is shot down during an engagement, the helicopter company does not automatically dissolve its joint intention to execute its mission; that would waste the team's jointly invested efforts in the mission and render the company highly ineffective in combat. Therefore, if a team member μ is known to drop its normal achievement goal, the JPG's dissolution is modified to be conditional on: (i) μ's role being critical to the continuation of the joint intention (as discussed in the next section); or (ii) pre-specified conventions. However, if μ communicates achievement, unachievability or irrelevance, then the JPG is dissolved as usual.

3.2 Complex Teams, Individual Roles and Failures

While not defined in terms of individual intentions, a joint intention leads individuals or subteams in the team to intend to do their "share" (role) of a team activity (subject to the joint intention remaining valid)[3]. In our work, a role constrains an individual or a subteam to undertake certain activities in service of the joint intention, and the role may vary with the joint intention.

One key issue here is that in complex teams, that involve multiple subteams, the success or failure of an individual's role performance does not directly determine the achievement or unachievability for the team's joint venture. As a result, an individual may succeed or fail in its role, yet communication may not necessarily result. Hence agents must communicate their role success or failures to other participants (should others be banking on this role performance). Furthermore, since agents may be unable to communicate (e.g., because costs exceed benefits), team members must track other agents' role performance. Based on information about others' role non-performance, team members can determine the viability of the team's joint intention or their own role. Two heuristics may be used:

1. *Critical expertise heuristic*: If the success of the team's joint intention is solely dependent on the role of an individual agent, then the agent's role non-performance (failure) implies that the team's joint intention is unachievable.

2. *Dependency heuristic*: If an agent's own role performance is dependent on the role of the non-performing agent, then the agent's own role performance is unachievable.

4 Implementing the Modified Joint Intentions Framework

To implement the modified joint intentions framework the concept of *team operators* has been defined. For the team Θ, a team operator OP will be denoted as $\boxed{\text{OP}}_{\Theta}$. The usual operators as seen in Figure 2 will henceforth be referred to as individual operators. As with individual operators, team operators also consist of: (i) precondition rules for selection; (ii) application rules (complex team operators will lead to subgoals); and (iii) termination rules. However, unlike individual operators, team operators encode the expressiveness and commitments of joint intentions.

4.1 Team Operators: Expressiveness

Team operators express a team's joint activity rather than an agent's own activity. Thus, while individual operators apply to an agent's own state, a team operator applies to a "team state". The team state is an agent's (abstract) model of the team's mutual beliefs about the world, which include identities of members in the team, information about their joint tasks etc. For instance, for a helicopter company, the team state may include the routes to fly to the battle position. Figure 4 shows the new operator hierarchy of helicopter pilot agents where operators shown in boxes such as $\boxed{\text{Engage}}_{\Theta}$ are team operators (the non-boxed ones are individual operators). These team operators are not tied to any specific number of agents within a team.

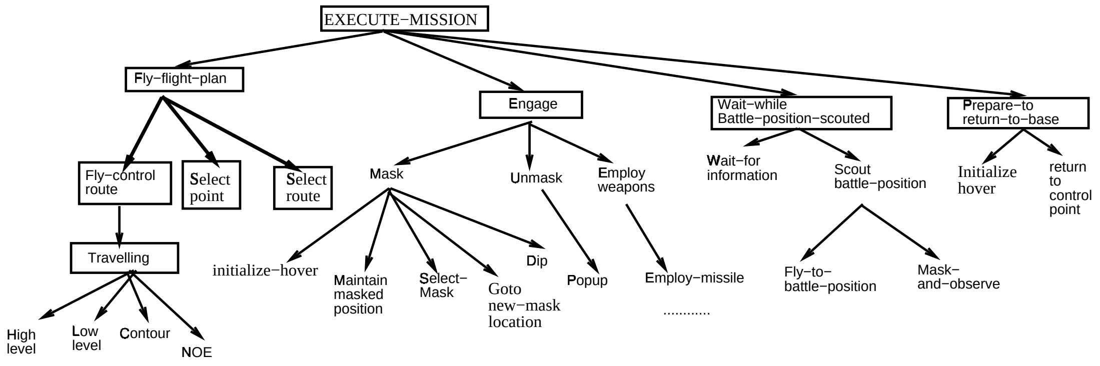

Figure 4: A portion of the new operator hierarchy, executed by an individual pilot agent.

To establish a joint intention $\boxed{\text{OP}}_{\Theta}$, each team member individually selects that team operator. Typically, this selection is automatically synchronized, since the selection is constrained by the team state (the team operator's preconditions must match the team state). Thus, since agents track their team state, visually and also via communication for terminating the previous team operator, it is usually unnecessary to explicitly communicate prior to the selection of the next team operator. This is as it should be for well-understood team activities. Note that in general, the subgoal of a team operator may lead to either a team operator or an individual operator to be applied. Thus, a joint intention may lead to either another joint intention or to individual intentions in a subgoal (subject to the parent joint intention remaining valid). For instance, while the children of $\boxed{\text{Engage}}_{\Theta}$ are all individual operators, the children of $\boxed{\text{Fly-flight-plan}}_{\Theta}$ are all team operators.

4.2 Team operator: Communication

Once selected, a team operator can only be terminated by updating the team state (mutual beliefs) to satisfy the team operator's termination rules. Updating the team state may lead to a communicative goal. In particular, if an agent's private state contains a belief that makes a team operator achieved or unachievable, and such a belief is absent in its team state, then it automatically creates a communicative goal, i.e., a communication operator. When executed, this operator leads the agent to broadcast the information to the team. For instance, suppose the team is executing $\boxed{\text{Engage}}_\Theta$, which is achieved if the team state contains the belief *Completed(Engagement)*. Now, if a (commander) pilot agent's own state contains *Completed(Engagement)*, and this is absent in its team state, then a communication operator is proposed to inform team members (the commander cannot just head back to home base alone).

To alleviate communication costs, certain safeguards are already built into the proposal of a communication operator. In particular, a communication operator is not generated if the private belief does not contribute to the achievment or unachievability of any active team operator, or if the team state is already updated,i.e., the team is already aware of the belief. Furthermore, based on the modifications discussed in Section 3.1, even if a communication operator is proposed, it is not implemented immediately. Instead, the agent first evaluates the cost and benefits of the communicative operator. For instance, if radio is the current means of communication, and if the mission requires radio silence, communication over the radio is prohibited. An agent instead attempts to reduce communication costs via alternative communication methods, e.g., travelling to personally deliver the message. If the agent finally satisfies its communicative goal, the sender and the receivers then update their team state (we assume that communicated information reaches other agents securly). This then causes the team operator to be terminated (either because it is achieved or unachievable). If a high-level team operator is achieved or unachievable, its children are automatically assumed irrelevant.

4.3 Team Operators: Roles, Failures and Recovery

For team operators, roles are instantiated via suboperators in the operator hierarchy. If an $\boxed{\text{OP}}_\Theta$ has $\mathcal{R}$ roles, denoted $\boxed{\text{OP}}_\Theta < \gamma_1, ..., \gamma_R >$, then Θ's R sub-teams, $\sigma_1 ... \sigma_R$, must undertake each of these roles. Many team operators, however, can be defined via multiple role combinations. For instance, $\boxed{\text{Engage}}_\Theta$ may be performed by anywhere between two to eight agents, some of them attack helicopters and some scouts. A separate representation of $\boxed{\text{OP}}_\Theta < \gamma_1, ..., \gamma_R >$ for each role combination would result in a large number team operators.

To alleviate this concern, constraints are specified to only implicitly define role combinations. For instance, for $\boxed{\text{Engage}}_\Theta$, the constraints specify that the allowable role-performing subteams are individual team members, i.e., the role performing subteam σ_i = I where I$\in$ Θ; without any constraints on the number of participants. Each agent instantiates the constraint relevant to itself, to know if it is expected to act alone or as part of a subteam. The actual role an agent undertakes is based on this allowable subunit, and any static specification of the subunit's role in the current situation (e.g., an agent may be specified to be a scout). This role specification is in turn based on the subunit's or individual's capability. For a company of helicopters, a specific individual may be the commander (capability depends on the chain of command), a scout (capability depends on training), or the leader of a formation (every team member possess this capability).

As mentioned in Section 3.2, it is useful for an agent to monitor other agents' role performance. This is accomplished in one of three ways. First, the other agent may itself communicate. Second, it is possible to track the other agent's role performance, via techniques such as RESC[23, 21], that dynamically infer other agents' higher-level goals and behaviors from observation of that agents actions. Given its expense, however, such detailed tracking is performed selectively — instead, an agent often only monitors the participation of other team members. Third, other heuristics can also be applied, e.g., an agent cannot perform two conflicing roles simultaneously. Thus, if a scout is scouting the battle position, it cannot participate in any other role at the holding area (e.g., to fly in formation).

The following describes the overall recovery algorithm, should an agent determine that $\mu \in \Theta$ is simply unable to perform any role (e.g., μ's helicopter crashes):

1. Let $\mathcal{R}$ = {r$_1$,...r$_N$} be the set of currently known roles of μ.
2. For each $\boxed{\text{OP}}_\Theta$ in currently active hierarchy and for each $r_i \in \mathcal{R}$ apply *critical expertise heuristic* to determine if $\boxed{\text{OP}}_\Theta$ unachievable.
3. If some $\boxed{\text{OP}}_\Theta$ unachievable, due to critical role r_c
   (a) Terminate $\boxed{\text{OP}}_\Theta$ and its active children.
   (b) If self capable of performing r_c, Communicate takeover of r_c to Θ; Re-establish $\boxed{\text{OP}}_\Theta$.
   (c) If self incapable of performing r_c, Wait for another agent to takeover r_c; Re-establish $\boxed{\text{OP}}_\Theta$. If wait too long, $\boxed{\text{OP}}_\Theta$ unrepairable.
4. For each $r_i \in \mathcal{R}$ apply *dependency heuristic* to determine if unachievable; apply domain-specific recovery strategies.
5. For all $r_j \in \mathcal{R}$, $r_j \neq r_c$, If self capable of performing r_j, Communicate takeover of r_j to Θ.
6. While μ disabled from performing any roles, check every future $\boxed{\text{OP}}_\Theta$ via critical expertise heuristic.

One key reason this recovery procedure works is the explicit representation of team operators. In particular, step 2 applies the *critical expertise heuristic*. To operationalize this heuristic, the agent compares the achievement condition of an $\boxed{\text{OP}}_\Theta$ with the achievement condition of μ's role. If identical, μ was solely responsible for achievement of $\boxed{\text{OP}}_\Theta$, and hence $\boxed{\text{OP}}_\Theta$ is now unachievable. Thus, if μ is a scout, this test indicates that it is critical to the scouting of the battle position. In Step 3-a, the agent terminates $\boxed{\text{OP}}_\Theta$ only if μ plays a critical role in $\boxed{\text{OP}}_\Theta$. In step 3-b, the agent attempts to substitute itself for μ's critical role if capability exists, or else it waits for someone else to fill in the role (step 3-c). Otherwise the implicated $\boxed{\text{OP}}_\Theta$ is irreparable.

In step 4, the agent attempts to recover from any individual operator dependencies (step 4). Here, to operationalize the *dependency heuristic,* the agent checks the achievement condition of its own role for μ's role. For instance, if an agent is to trail μ in formation, its achievement depends on μ. Non-critical roles are examined later, as they may be critical in the future (step 5). It is possible that one agent does not possess all of μ's capabilities, and hence may takeover only one of μ's roles, while other agents takeover μ's other roles. Not all of μ's roles may be known immediately; and hence any new operator is also checked for critical dependency on μ (step 6).

To see the above procedure in action, consider a company of five helicopters, Cheetah421 through Cheetah425, with the role and capabilities as shown:

Current roles:
Cheetah421 <— Commander, Scout
Cheetah422, Cheetah423,Cheetah424, Cheetah425 <— Attack
Current capabilities:
Cheetah421,Cheetah423 <— Scout
Cheetah422, Cheetah423,Cheetah424, Cheetah425 <— Attack
Chain of command: Cheetah421->Cheetah422->Cheetah423->Cheetah424->Cheetah425

Suppose, the team is currently executing $\boxed{\text{wait-while-bp-scouted}}_\Theta$. In service of this team operator, the scout (Cheetah421) is moving forward to scout the battle position, while the rest of the company is waiting at the holding area. Now if the scout crashes (as in Item 1 in Figure 3), $\boxed{\text{wait-while-bp-scouted}}_\Theta$ is deemed unachievable (critical expertise heuristic). Two changes will then take place. First, Cheetah423 will take over the critical role of the scout — it has the capability of becoming a scout. This enables the $\boxed{\text{wait-while-bp-scouted}}_\Theta$ operator to be re-established for execution. Next, Cheetah422, the next in command, will replace Cheetah421 as the commander.

5 Experimental Results

Agents based on our new approach each currently contain 1000 rules, with roughly 10% rules dedicated our explicit model of teamwork. This new implementation addresses three basic types of problems seen in our previous implementation:

- Recovery from incapabilities of key individuals, such as a commander or a scout (e.g., addresses Item 1, Figure 3).
- Better communication and coordination within the team, as members recognize responsibilities (e.g., addresses Items 2 and 3, Figure 3).
- Improved tracking of own team state due to improved expressiveness (e.g., addresses Item 4, Figure 3); also possible to track team's high-level goals and behaviors, not possible before.

Figure 5 illustrates that our current implementation provides significant flexibility in the level of coordination among team members. The figure attempts to plot the amount of coordination among team members (y-axis) over simulation time (x-axis). The percentage of team operators in a pilot agent's operator hierarchy (which consists of team and individual operators) is a rough indicator of the amount of coordination. In particular, a lower percentage of team operators implies a higher percentage of individual operators and hence low coordination among members; while a higher percentage of team operators indicates tighter coordination. Time is measured in simulation cycles, with 9475 cycles in this run.

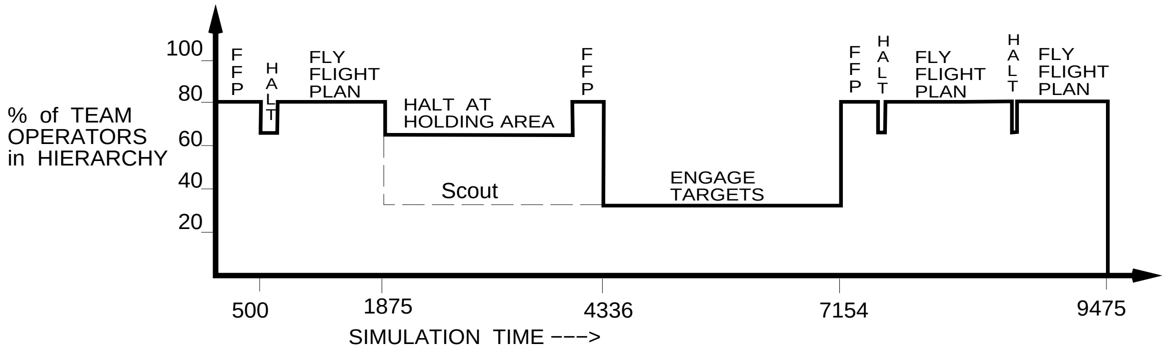

Figure 5: Percent team operators in an individual's operator hierarchy (FFP = Fly Flight Plan).

The varying percentage of team operators over the run indicates the flexibility in the level of coordination. Thus, for the first 500 cycles, when the agents are flying a flight plan (FFP) in close formation, they are tightly coordinated, an individual's operator hierarchy has 80% team operators. For the next 50 cycles, the company halts, and then resumes flying its flight plan. At cycle 1875, the company reaches the holding area, where the scout files forward to scout the battle position — the scout's percentage is shown separately by a dashed line. Basically, the scout is now only loosely coordinating with the rest of the company (33% team operators). After scouting, the company moves the battle position at cycle 4336, and until cycle 7154, engages targets. The 33% team operators in engaging targets indicate that the team members are to a large extent acting independently. Nonetheless, the team operator percentage is never zero, i.e., these agents never act completely alone. Later the company returns to base.

Figure 6 illustrates the reduction in communication due to our modifications to the joint intentions framework. It shows results from a single test run of our implementation. Figure 6-a projects percentages of operators, had the agent worked with the original joint intentions framework. In this case, there are 25% team operators; and among the approx 75% individual operators, there are 25% communication operators and the rest execute the agents' actions. Figure 6-b shows the percentage from an actual run with the modified joint intentions framework. Communication percentage decreases more than 10-fold (just about 2% on communication). Instead, there is more emphasis on agent- and team-tracking, performed using RESC[23, 21], with about 8% operators.

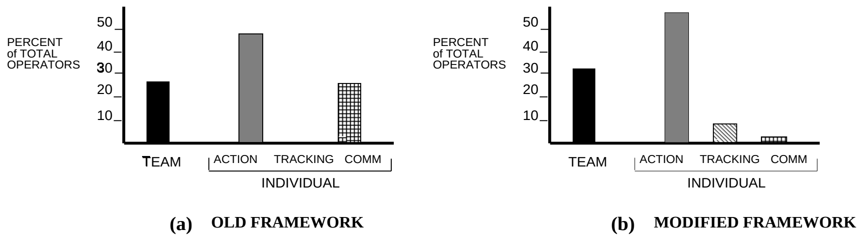

Figure 6: Reduction in percentage of communication operators.

6 Related Work

Few other research efforts have implemented theories of joint action. Jennings's implementation of the joint intentions framework in an industrial multi-agent setting is one notable exception[11]. Huber and Durfee describe a similar implementation, although in a smaller scale testbed[8]. There are several key differences in our work. First, in both these efforts, agents' collaborative activity appears to involve a two level hierarchy of a joint goal and a joint plan, with individuals engaged in specific roles in the plan. When the joint goal is accomplished, the collaborative activity is terminated. In contrast, our work focuses on complex, long-term team activities, involving the execution of a dynamically changing team operator hierarchy. A high-level mission leads to the execution of a whole variety team operators. It thus becomes essential to maintain and track an explicit team state, and manipulate it via team operators — else agents will lose track of the next team action. Second, the above efforts typically involve two-three agents in the joint intention. The scaleup from two-three agent to five-eight agent per teams (as in our work) creates new possibilities. More specifically, even if a single agent is incapacitated, the team operator hierarchy does not completely fall apart. However, agents have to explicitly check if lower-level team operators are unachievable, and recover from failures. Recovery is important, else the entire team effort will go to waste. Finally, in [11] issues of communication risk are not considered (although they are considered in [8]).

Our recent work on team tracking[21] — which involves inferring other team's joint goals and intentions based on observations of their actions — is the predecessor to the work reported here. However, given its focus on tracking other teams, issues such as communication, recovery from unachievable team operators were all explicitly excluded from consideration. The domain of focus there was tracking the behaviors of a team of enemy fighter jets.

7 Summary and Discussion

In a variety of dynamic multi-agent environments currently under development, achieving flexibility in teamwork is critical[7, 22, 1, 13]. Yet, given the uncertainity in such domains, preplanned coordination cannot sustain such flexible teamwork. To alleviate this problem, we have provided individual agents with an explicit representation of team goals and plans, and an underlying explicit model of team activity, which has already substantially improved agents' flexibility in their teamwork. Further contributions of this paper include: (i) Detailed illustration of an *implementation* of the modified joint intentions framework[3] in a real-world multi-agent domain; (ii) key modifications to the joint intentions framework to reflect important constraints in the domain; (iii) introduction and implementation of team operators(reactive team plans); (iv) techniques for recovery from failure of team activities. As an important side-effect, agent development has speeded up, since once agents are equipped with such a model of teamwork, the knowledge engineer can specify higher-level *team plans*, and let the individual agents reason about the coordination activities and recovery.

Our work focused on one real-world domain, with key charachteristics that appear representative of other real-world domains; and thus the lessons learned appear to have wider significance. The key lessons here are that as we build agent teams for increasily complex multi-agent systems, agents should be provided (i) explicit representations of team activities, and more importantly (ii) some core commonsense knowledge of teamwork, separate from the agent's domain-level expertise (e.g., helicopter tactics). These lessons appears applicable to other dynamic multi-agent domains, including other applications of the simulation technology described here such as training for (natural) disaster relief, medical emergencies etc. Indeed, to test these lessons, we have begun implementing this framework for players in the RoboCup virtual soccer tournament[13].

References

[1] J. Bates, A. B. Loyall, and W. S. Reilly. Integrating reactivity, goals and emotions in a broad agent. Technical Report CMU-CS-92-142, School of Computer Science, Carnegie Mellon University, May 1992.

[2] R. B. Calder, J. E. Smith, A. J. Courtemanche, J. M. F. Mar, and A. Z. Ceranowicz. Modsaf behavior simulation and control. In *Proceedings of the Conference on Computer Generated Forces and Behavioral Representation*, 1993.

[3] P. R. Cohen and H. J. Levesque. Teamwork. *Nous*, 35, 1991.

[4] R. Davis. Expert systems: where are we? and where do we go from here? *AI Magazine*, 3(2), Spring 1982.

[5] J. Firby. An investigation into reactive planning in complex domains. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1987.

[6] B. J. Grosz and C. L. Sidner. Plans for discourse. In *Intentions in Communication*, pages 417–445. MIT Press, Cambridge, MA, 1990.

[7] B. Hayes-Roth, L. Brownston, and R. V. Gen. Multiagent collaobration in directed improvisation. In *Proceedings of the International Conference on Multi-Agent Systems (ICMAS-95)*, 1995.

[8] M. Huber and E. Durfee. On acting together: Without communication. In *Proceedings of the AAAI Spring Symposium on Reasoning about Mental states*, 1995.

[9] F. F. Ingrand, M. P. Georgeff, , and A. S. Rao. An architecture for real-time reasoning and system control. *IEEE EXPERT*, 7(6), 1992.

[10] N. Jennings. Commitments and conventions: the foundation of coordination in multi-agent systems. *The Knowledge Engineering Review*, 8, 1994.

[11] N. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75, 1995.

[12] D. Kinny, M. Ljungberg, A. Rao, E. Sonenberg, G. Tidhard, and E. Werner. Planned team activity. In C. Castelfranchi and E. Werner, editors, *Artificial Social Systems, Lecture notes in AI 830*. Springer Verlag, New York, 1992.

[13] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. Robocup: The robot world cup initiative. In *Proceedings of IJCAI-95 Workshop on Entertainment and AI/Alife*, 1995.

[14] Y. Kuniyoshi, S. Rougeaux, M. Ishii, N. Kita, S. Sakane, and M. Kakikura. Cooperation by observation: the framework and the basic task pattern. In *Proceedings of the IEEE International Conference on Robotics and Automation*, May 1994.

[15] H. J. Levesque, P. R. Cohen, and J. Nunes. On acting together. In *Proceedings of the National Conference on Artificial Intelligence*. Menlo Park, Calif.: AAAI press, 1990.

[16] A. Newell. *Unified Theories of Cognition*. Harvard Univ. Press, Cambridge, Mass., 1990.

[17] K. Pimentel and K. Teixeira. *Virtual reality: Through the new looking glass*. Windcrest/McGraw-Hill, Blue Ridge Summit, PA, 1994.

[18] S. Rajput and C. R. Karr. Cooperative behavior in modsaf. Technical Report IST-CR-95-35, Institute for simulation and training, University of Central Florida, 1995.

[19] A. S. Rao, A. Lucas, D. Morley, M. Selvestrel, and G. Murray. Agent-oriented architecture for air-combat simulation. Technical Report Technical Note 42, The Australian Artificial Intelligence Institute, 1993.

[20] P. S. Rosenbloom, J. E. Laird, A. Newell, , and R. McCarl. A preliminary analysis of the soar architecture as a basis for general intelligence. *Artificial Intelligence*, 47(1-3):289–325, 1991.

[21] M. Tambe. Tracking dynamic team activity. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, August 1996.

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

[23] M. Tambe and P. S. Rosenbloom. RESC: An approach for real-time, dynamic agent tracking. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1995.

[24] M. Tambe, K. Schwamb, and P. S. Rosenbloom. Building intelligent pilots for simulated rotary wing aircraft. In *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, May 1995.

[25] G. Tidhar, M. Selvestrel, and C. Heinze. Modeling teams and team tactics in whole air mission modelling. Technical Report Technical Note 60, The Australian Artificial Intelligence Institute, 1995.