In *Proceedings of the 15th National Conference on Artificial Intelligence* (AAAI-98)

What is Wrong With Us?
Improving Robustness Through Social Diagnosis

Gal A. Kaminka and **Milind Tambe**
Information Sciences Institute and Computer Science Department
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292
{galk, tambe}@isi.edu

Abstract[1]

Robust behavior in complex, dynamic environments mandates that intelligent agents autonomously monitor their own run-time behavior, detect and diagnose failures, and attempt recovery. This challenge is intensified in multi-agent settings, where the coordinated and competitive behaviors of other agents affect an agent's own performance. Previous approaches to this problem have often focused on single agent domains and have failed to address or exploit key facets of multi-agent domains, such as handling team failures. We present SAM, a complementary approach to monitoring and diagnosis for multi-agent domains that is particularly well-suited for collaborative settings. SAM includes the following key novel concepts: First, SAM's failure detection technique, inspired by social psychology, utilizes other agents as information sources and detects failures both in an agent and in its teammates. Second, SAM performs social diagnosis, reasoning about the failures in its team using an *explicit model of teamwork* (previously, teamwork models have been employed only in prescribing agent behaviors in teamwork). Third, SAM employs *model sharing* to alleviate the inherent inefficiencies associated with representing multiple agent models. We have implemented SAM in a complex, realistic multi-agent domain, and provide detailed empirical results assessing its benefits.

Introduction

Attaining robustness in face of uncertainty in complex, dynamic environments is a key challenge for intelligent agents (Toyama and Hager 1997). This problem is exacerbated in complex multi-agent environments due to the added requirements for communication and coordination. Example domains include virtual environments for training (Tambe et al. 1995), robotic soccer (Kitano et al. 95), potential multi-robotic space missions, etc. The inherent explosion of state space complexity in these dynamic environments inhibits the ability of any designer, human or machine (i.e., planners), to specify the correct response in each possible state in advance (Atkins et al. 1997). For instance, it is generally difficult to predict when sensors will return unreliable answers, communication messages get lost, etc. The agents are therefore presented with countless opportunities for failure, and must autonomously monitor and detect failures in their run-time behavior, then diagnose and recover from them, i.e., agents must display *post-failure robustness* (Toyama and Hager 1997).

Previous approaches to monitoring and diagnosis (e.g., Doyle et al. 1986, Williams and Nayak 1996) have often focused on a single agent that utilizes designer-supplied information, either in the form of explicit execution-monitoring conditions, or a model of the agent itself. This information allows the agent to compare its *actual* behavior with the *ideal* behavior to detect failures. While powerful in themselves, these methods have several limitations in multi-agent dynamic environments.

First, these approaches are geared towards detecting and diagnosing failures in a single agent's own behaviors. They do not consider failures in other agents even when those affect the agent's own performance. For instance, a teammate's failure can change the ideal behavior expected of an agent, but this can only be known if the agent can detect teammates' failures.

Second, these single-agent approaches cannot capture team-level failures, where the failures may not be all in a single individual, but rather are distributed among a number of agents in a team. In this case diagnosis and recovery imply not only correcting the individual failure, but also re-establishing coordination at the team-level.

Third, the single-agent perspective in these approaches prevents them from utilizing other agents as sources of knowledge to compensate for possible failures in some of an agent's own sensors, i.e., to use whatever information is sensed about other agents to infer the failing sensors' results[2]. For example, a driver may not see an obstacle on the road, but if s/he sees another car swerve, s/he can infer the presence of the obstacle.

Fourth, these previous approaches are hard to scale up to complex, multi-agent environments. In particular, since agents in such environments adjust their behavior flexibly to respond to their actual circumstances, it becomes increasingly hard to specify the correct behavior. For instance, specifying a target range for monitoring the velocity of a car becomes difficult if we are to take into account some of the possible responses of the driver, e.g., acceleration beyond the speed limit or slowing down to

[2] This relies on the reliability sensors recognizing the other agents. If all sensors fail, little can be done.

avoid hitting another car.

To alleviate such limitations of the existing approaches, we have developed SAM (Socially Attentive Monitoring), a new approach to failure detection, diagnosis and recovery. SAM *complements* existing approaches by addressing the difficulties and exploiting the opportunities in multi-agent environments. It is particularly relevant for collaborative (teamwork) settings, that are ubiquitous in multi-agent environments. SAM allows detection of failures in the monitoring agent and its peers by a technique inspired by Social Comparison Theory (Festinger 1954). The key idea is that agents compare their own behavior, beliefs, goals, and plans to those of other agents, reason about the differences in belief and behavior (not necessarily imitating the others), and draw useful conclusions regarding the correctness of their own actions or their peers'. While information about other agents' beliefs can be obtained via communication, such communication can be a significant overhead or risk (in hostile environments). Instead, SAM uses plan recognition to infer other agents' beliefs, goals, and plans from their observable actions (communicating only if necessary and feasible). Thus, SAM would be applicable in the car swerving example discussed above. To reduce inefficiency in keeping track of other agents' beliefs/plans, SAM utilizes *model sharing* for efficient reasoning with such models.

Upon detecting a possible failure, SAM performs diagnosis using an *explicit teamwork model* to establish the exact difference in beliefs between the agents and the significance of the difference. Explicit teamwork models have begun to be used in multi-agent domains for teamwork flexibility (Jennings 1995; Tambe 1997). SAM exploits these models--and the guarantees they provide-- in a novel way, running them in reverse for diagnosis. A detailed diagnosis enables SAM to recover.

SAM complements previous work on execution monitoring and diagnosis by alleviating key weaknesses mentioned above. First, social comparison allows agents to detect failures in their peers' behavior, diagnose the failures, and recover either by adjusting own behavior or influencing the failing agents (e.g., by communicating with the failing agents). Second, SAM utilizes an explicit teamwork model to facilitate diagnosis and recovery at the team-level (re-establishing team coordination), and not just at the individual level. Third, SAM enables an agent to compensate for some of its failures to sense the environment directly, by sensing other agents' behaviors, from which SAM infers missing information about the environment. Finally, SAM complements the models or conditions normally specified by the designer with information gained from other agents. These other agents behave flexibly in their environment and thus effectively provide a model that matches the particular state the agent is facing. However, SAM has to address the possibility that these other agents may be behaving erroneously.

Motivation and Examples

The motivation for our approach comes from our application domain which involves developing automated pilot agents for participation in a commercially-developed battlefield simulation (Tambe et al. 1995). This real-world environment is complex and dynamic, with uncertainties such as unscripted behaviors of other agents, unreliable communications and sensors, possibly incomplete mission and task specifications, etc. These qualities present the agents with never-ending opportunities for failure, as anticipation of all possible internal and external states is impossible for the designer. Two example failures may serve to illustrate. The first failure involved a scenario where a team of three helicopter pilot agents was to fly to a specified landmark position. Having reached this position, one of the team members, whose role was that of a *scout*, was to fly forward towards the enemy, verifying its position. The scout's two teammates (role: *attackers*) were to land and wait for its return to the specified position. All of the pilot agents were explicitly provided conditions to monitor for the landmark. However, due to an unexpected sensor failure, one of the attackers failed to sense the landmark marking the waiting position. So while the other attacker correctly landed, the failing attacker continued to fly forward with the scout, following the original plan which called for flying in formation! The failing attacker had clearly seen that the other attacker had landed, but it did not use this information to infer the position of the landmark. Furthermore, the other attacker and the scout did not complain to the failing attacker about its failure. In a second example, a similar team of three helicopters was to take off from the home base and head towards their battle position. One of the agents unexpectedly did not receive the correct mission specification, so while two of the agents began to fly out as planned, the failing agent kept hovering in place at the starting position indefinitely. Again, none of the agents complained about the unusual performance of the team.

We have collected dozens of such failure reports during the last two years. While it is generally easy for the human designer to correct these failures once they occur, it is hard to anticipate them in advance. These failures occur despite significant development and maintenance efforts. Given the complexity of the dynamic environment, predicting all possible states and all possible interactions is impossible. Furthermore, these failures are not negligible. Rather, they are very obvious (to the human observer) catastrophic failures, for both individual agents and the team. In the second example above, not only was the single agent stuck behind unable to participate in the mission, but the remaining agents were unable to carry out the mission by themselves.

Furthermore, these failures are not due to a lack of domain expertise, as the domain experts expect some common sense handling of such failures even in the most structured military procedure. Indeed, by exercising social common sense, an agent may at least detect that something may be wrong. Social clues, such as (in the examples above) noticing that teammates are leaving while the agent is hovering in place, or that a team-member has landed while the team was flying in formation, would have been sufficient to infer that something may be wrong.

Social Monitoring: Detecting Failures

SAM is composed of three processes: (i) a failure detection process, which involves comparison with peers, in particular teammates, to detect the possibility of failure, (ii) a diagnosis process to confirm the detected failure and perform detailed diagnosis, and (iii) a recovery process. We begin in this section by describing SAM's failure detection process, which is inspired by Social Comparison Theory (Festinger 1954) from social psychology. The first three axioms of this theory are as follows: (1) Every agent has a drive to evaluate its opinions and abilities; (2) If the agent can't evaluate its opinions and abilities objectively, then it compares them against the opinions and abilities of others; (3) Comparing against others decreases as the difference with others increases.

The numerous failure reports we have collected empirically demonstrate the very real need of agents in dynamic, unpredictable domains to evaluate themselves by monitoring their execution (first axiom). Current approaches emphasizing the designer as a source of information against which to compare the agent's performance fit naturally under the title of objective sources for the agent's self-evaluation. SAM's detection technique is inspired by the remaining parts of the axioms--allowing the agent to compare its own abilities and opinions (i.e., behavior, beliefs, and goals) to those of others, and considering the weight put on these comparisons (third axiom).

SAM's monitoring technique is an operationalization of this descriptive social comparison process. To detect possible failures, SAM compares the monitoring agent's own state and the state of other agents -- where an agent's state may include its beliefs, goals, behaviors, etc. However, as implied by the third axiom of social comparison theory, differences with other agents are meaningful only to the extent that the other agents are *socially similar*. If other agents' states are expected to be dissimilar to the agent's state, they may be unable to contribute relevant information towards the monitoring of the agent's own performance. Also, hostile agents may intentionally want to use deception in order to influence the agent's decision making to advance their own agendas.

To address this issue, SAM currently limits its comparison to the agent's teammates only, as these tend to work on common goals and related sub-plans, and can be assumed to be non-hostile (given the ubiquity of teamwork in multi-agent domains, this is not a limiting assumption). Differences with team members' states may imply a *possible* failure in either agent. For instance, in the second helicopter example above, the agent left hovering at the starting point could detect a possible failure by comparing its own state with teammates' states.

SAM's capabilities may differ depending on what state information about teammates is compared, e.g., internal beliefs and goals vs. observable behavior. In general, the information compared should satisfy a trade-off between two criteria: (i) For efficiency, only limited information should be maintained about other agents and compared, and (ii) For maximized monitoring capabilities, the information should capture as much as possible of the agents' beliefs, goals, and internal control processes, since we can only hope to detect discrepancies in the actual agent-attributes we are comparing.

Our agents' design is based on reactive plans (operators) (Firby 1987, Newell 1990), which form a decomposition hierarchy that controls each agent. Operator hierarchies provide a good trade-off between the criteria considered above, as they are both compact enough to be reasoned about efficiently, while being central to the agent behavior (capturing its decision process). We therefore chose operator hierarchies for our comparison purpose. Figure 1 presents a small portion of such a hierarchy. Each operator in the hierarchy has preconditions for selecting it, application conditions to apply it, and termination conditions. The design of the hierarchical plans uses the STEAM framework (Tambe 1997) for maintaining an explicit model of teamwork. Following this framework, operators may be team operators (that explicitly represent the joint activities of the team) or individual (specific to one agent). In Figure 1, boxed operators are team operators, while other operators are individual. The filled arrows signify the operator hierarchy currently in control, while dotted arrows point to alternative operators which may be used. In the figure, the agent is currently executing the execute-mission team operator which is its highest-level team operator, and has chosen (jointly with its team) to execute the fly-flight-plan operator, for flying with the team through the different mission-specified locations.

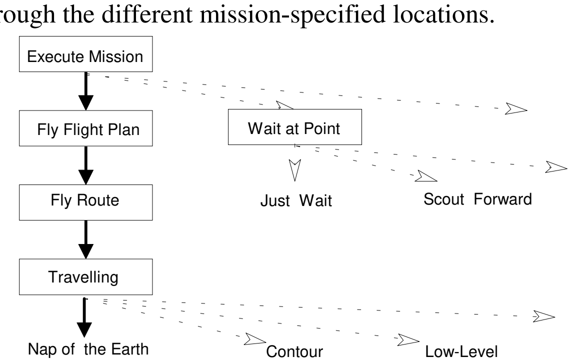

Figure 1. An example operator hierarchy.

To compare its own operator hierarchy with other agents' hierarchies, an agent must acquire knowledge about the others. Such knowledge can be acquired in two ways: it can be communicated by the other agents or it can be inferred by the monitoring agent based on its observation of the other agents via plan recognition. The choice to prefer one method over the other (or use a combination of both) is dependent on the (i) cost and (ii) expected reliability of these methods, which changes across domains.

In many realistic domains, continuous communication involves significant cost, both in overhead and risk. For example, in our battlefield simulation domain, the cost of communications is very high, as the agents operate in a hostile environment and expose themselves to risks by communicating with each other. In contrast, the cost of plan recognition is relatively low, being mostly a computational cost rather than a survival risk. In addition, in a team context, plan recognition is often quite reliable due to assumptions that can be made about the behavior of other agents, since they are team members. Our estimates of reliability and the cost make plan recognition an attractive choice for acquiring knowledge of others.

We use the $RESC_{team}$ (Tambe 1996) method for plan-recognition, but different techniques may be used interchangeably, as long as they provide the needed information and representation. $RESC_{team}$ provides real-time plan recognition capabilities, constructing operator hierarchies (in the recognizing agent's memory) that correspond to the other agents' currently executing reactive operators. The monitoring agent therefore has unified access not only to its own hierarchies, but also to the (inferred) operator hierarchies of other team members, constructed by $RESC_{team}$. In figure 2 below, the left hierarchy is the hierarchy actually in control of the agent. The right one is inferred by $RESC_{team}$ to be currently executed by another agent.

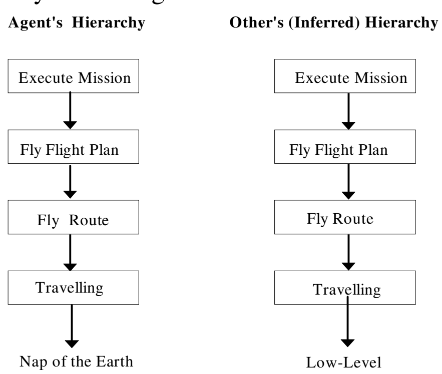

Figure 2. Two example hierarchies in the agent's memory.

Comparing the agents' operator hierarchies involves a top-down comparison of the operators in equal depths of the hierarchies (hierarchies of different lengths are also considered different). In figure 2, the difference that would be detected is between the two leaf nodes, since all operators above them are identical.

Any such difference indicates a possible failure, if teammates were supposed to be executing similar operators in the first place. Before we discuss further diagnosis of this failure (next Section), we address here one inherent inefficiency in this failure detection method--it potentially needs to keep track of the operator hierarchies of all other teammates. To alleviate this inefficiency, we rely on *model-sharing*--the use of shared hierarchies for team operators (Tambe 1996). Team operators at equal depths (the agent's own, and those inferred of others), which should be identical for all team-members, are shared in the agent memory. Thus, only individual operators are maintained separately. When team-operator differences occur (see next section), SAM unshares the hierarchies at the point at which they differ to allow reasoning about the differences to continue, without having to maintain separate hierarchies needlessly. Model sharing thus maintains conceptually different hierarchies, while limiting the actual computational overhead.

Social Diagnosis and Recovery

While SAM's detection process indicates possibilities of failures, its diagnosis process verifies the failure and generates an explanation for it utilizing social knowledge sources (other agents and the explicit teamwork model). These sources determine the expected similarity between the agents involved, and thus determine whether, and to what degree, the difference in operators is truly an indication of failure. In particular, differences can be detected at the team level (the monitoring agent and its teammates are not executing the same team operator), or individual level. The extent of the diagnosis and recovery depends on the type of difference detected.

Team-Operator Differences

In the case of team-operator differences, SAM's failure diagnosis is driven by explicit models of teamwork. Such models have recently emerged as a promising approach to provide greater flexibility in teamwork (Jennings 1995; Tambe 1997). In particular, they alleviate the need for domain-specific coordination plans.

The key idea in SAM is based on the observation that teamwork models specify how a team should work in general. Thus, tracing back through such a model can help confirm and diagnose team failures. In particular, teamwork models contain domain-independent axioms that prescribe general responsibilities for team members and the team. These axioms in turn have been derived from teamwork theories, such as joint intentions (Levesque et al. 1990) and SharedPlans (Grosz and Kraus, 1996). Our basic idea is to backchain through these axioms to diagnose the failure. For instance, one axiom of the joint intentions framework mandates that a persistent (committed) team goal cannot be abandoned unless there exists mutual belief in the team goal being irrelevant, achieved, or unachievable. Thus, if the agent discovers a goal has been abandoned by others, it can infer they believe mutual belief has been established in the goal's irrelevancy, achievement, or unachievability.

In our implementation, the agents utilize one such explicit model of teamwork, STEAM (Tambe 1997), for their collaborative execution of team operators. STEAM ensures that team operators are established jointly by the team via attainment of mutual belief in their preconditions, and terminated jointly by attaining mutual belief in the team-operator's termination conditions (either achievement, unachievability, or irrelevancy conditions). In theory, team operators must therefore *always be identical* for all team members. However, given the well recognized difficulty of establishing mutual belief in practice (Halpern and Moses 1990), differences in team operators unfortunately do occur. Furthermore, for security or efficiency, team members sometimes deliberately reduce communication, and inadvertently contribute to such team-operator differences. For instance, agents may assume that an external object in team members' visual range (such as a landmark) is visible to all, and may not communicate about it.

Given STEAM's guarantees that the team operator must always be identical, any team-operator differences detected by SAM therefore imply not only a possibility but a certainty of team coordination failure. Having established this certainty in the failure, SAM's team-level diagnosis next attempts to identify the exact differences between its own beliefs and its teammates' beliefs that have led them to execute different team operators--this aspect of diagnosis is key for recovery, and it proceeds as follows. First, given a difference between the monitoring agent's **T1** team operator and the other team-members' **T2** team operator, SAM infers that the entire team was initially executing **T1** (since no differences were detected earlier). However, now teammates have begun executing **T2**[3]. Therefore, SAM infers that the teammates believe that one or more of the disjunctive preconditions necessary for selection and execution of **T2** were satisfied. Furthermore, SAM infers that teammates believe that one or more of the disjunctive termination conditions of **T1** have been achieved. Typically, the intersection between these two sets of possible beliefs determines the actual set of beliefs that the teammates hold that are different at this point[4]. In addition, the teamwork model guarantees SAM that this is the only real difference that has led to the teammates' executing **T2**. Of course, this intersection idea can be applied to both team and individual operator differences, but it is of particular importance for team-level failures given the guarantees provided by the teamwork model.

In the example of the agent's failure to detect a key landmark, SAM infers that the other agents are carrying out the wait-at-point operator (one attacker lands, while the scout goes forward). Once this discrepancy is noted ("I am executing fly-flight-plan, they are executing wait-at-point"), SAM determines that since the other agents have terminated "fly-flight-plan" they have either met with an enemy, or reached the landmark. From their current choice of "wait-at-point" operator (whose preconditions include reaching the landmark), SAM infers that the teammates believe that they have indeed reached the landmark. Thus, given the guarantees of teamwork models, the difference in team operators is elaborated by the diagnosis process to infer specific differences in team members' beliefs.

[3] The case where the team did not switch but the monitoring agent did is also possible, but is not described here for brevity

[4] Empty intersection cases are a topic for future work.

SAM next continues to backchain through the teamwork model, attempting to glean further information about the belief difference. To begin with, it infers if this difference of beliefs among team members is one that causes the operator in question to be achieved, unachievable, or irrelevant. SAM can then diagnose the failure in further detail. For instance, in a different failure scenario, the scout was shot down but this was known only to some agents in the team. They then began to communicate as part of a replan procedure to take over the role of the dead scout. The agents that did not see the scout crash failed to understand why these messages were sent, and chose to ignore them, therefore causing a failure in coordination. However, using SAM, the monitoring agent indirectly determines that the scout is no longer functioning when it receives communication messages from its peers calling for a replan. In particular, SAM first infers that the other agents are executing a "replan" team-coordination operator. This triggers a difference with the agent's own "wait-at-point". SAM then infers that the preconditions for a replan have become true, which means that there was a failure in "wait-at-point" which made it unachievable (as opposed to achieved or irrelevant). The only possible reason for this is that the scout cannot complete its mission.

Recovery is greatly facilitated by better diagnosis. Currently, SAM's recovery assumes that the agent's perception is incomplete, but not inconsistent. For example, an agent's sensors may fail to detect the landmark (a "don't know" response), but would not erroneously say it is there when it isn't. Thus, SAM's diagnosis that teammates have come to believe in something which the monitoring agent does not know about (or vice versa) enables the monitoring agent to recover by adopting this belief (or in the reverse case, by letting others know about this belief). The above assumption is made only in the recovery stage, not in the detection or diagnosis stages, and removing it is a topic for future work.

In the example of the failure to detect the landmark, once the agent diagnoses the problem that the other agents have detected the landmark (making the "fly-flight-plan" achieved), it recovers completely by adopting the other agents' belief. This adoption of belief makes the preconditions for its own "wait-at-point" operator true, and it re-establishes mutual belief with the team, completely resolving the problem.

Individual Operator Differences

In service of team operators different agents may work on different individual operators. Thus, a difference in individual operators may not necessarily signify failure. Individual operators do not carry with them the responsibilities for mutual belief that team operators do, and also none of the guarantees provided by a teamwork model. Therefore, SAM consults additional information about the agents causing the difference which can help in determining whether the difference is justified or not, prior to embarking on the diagnosis of the exact set of differences in beliefs. Agents working towards similar goals have similar *social roles* or *status.* For example, in a soccer game there are field players and a goalie which have different roles within the team. Agents with similar roles serve as better sources of information for plan-execution monitoring than agents with different roles. In some cases, SAM may choose to completely eliminate some agents which are too socially dissimilar from further consideration even at the detection level, so that differences in choice of individual operators will no longer signify possible failures. At the team level, however, these agents are still very much considered.

SAM will therefore choose to fully diagnose only differences with agents of similar role/status, assigning appropriate discounted weight to the explanation generated. The example where a failing agent was stuck in place while its team-members have taken off and were flying away serves to illustrate. Here, the agent compares its chosen method-of-flight operator to the method-of-flight chosen by its comrades. This comparison is meaningful since at least one of the comrades has a similar role and status. The comparison indicates to the failing agent that it is not acting like its team-mates - that a failure may have occurred (see leaf operators in Figure 2). The process of diagnosis here results in SAM's inferring that the reason for the difference is that the mission specification of the flight-method (which is a precondition for selecting a particular method) is different between the agents. However, it is highly improbable that two different mission specifications were intentionally sent out to two agents in a team (domain-specific knowledge). Therefore, the monitoring agent communicates with teammates and their commander to possibly resolve this problem.

Results and Evaluation

Our agent, including SAM, is implemented completely in the Soar integrated AI architecture (Newell 1990). About 1200 rules are used to implement the agent, including the military procedures, teamwork capabilities (STEAM), and plan-recognition capabilities ($\text{RESC}_{\text{team}}$). Approximately 60 additional rules implement SAM, forming an add-on layer on top of the procedures making up the agent.

SAM can resolve a significant portion of the team and individual failures in our collected failure reports, too numerous to discuss here in detail. We have therefore chosen to illustrate SAM's strengths and limitations via systematic experimentation with variations of the landing point failure described above. Here, we systematically tested SAM's capabilities in all possible permutations of the original scenario, i.e., all possible pairings of agents' failures and roles. To make things tougher, only a single agent deployed SAM. Thus, in our experiments SAM had to detect and diagnose failures not only in the monitoring agent, but also in one or more of its teammates.

In the first set of experiments, our SAM-executing agent (called SAM in the tables below) plays its original role of an *attacker* (Table 1). The other agents (other-1 and other-2) also play their original role. For further variations in failure scenarios, we set up another set of experiments, where SAM plays the other role involved in the scenario, that of the scout (see Table 2).

The first column in the tables is experiment number. The next three columns note the three agent/roles in the experiments. Entries marked "Fail" indicate that in the given experiment (row), the agent in question has failed to detect the landing point, and thus continued execution of "fly-flight-plan". The "Detect" column marks "Yes" in experiments in which SAM was successful in detecting a difference in operators indicating possible failure (and "No" otherwise). The Diagnose column similarly indicates whether SAM's diagnosis procedure was successful at generating an explanation of the failure. Recovery was successful in all cases where a diagnosis was generated.

Exp #	**attack SAM**	attack other1	scout other2	Detect	Diagnose
1	-	-	-	N/A	N/A
2	Fail	-	-	Yes	Yes
3	-	Fail	-	Yes	Yes
4	Fail	Fail	-	No	No
5	-	-	Fail	Yes	Yes
6	Fail	-	Fail	Yes	Yes
7	-	Fail	Fail	Yes	Yes
8	Fail	Fail	Fail	No	No

Table 1. SAM - attacker, other-1 - attacker, other-2 - scout

Exp #	attack other1	attack other2	**scout SAM**	Detect	Diagnose
9	-	-	-	N/A	N/A
10	Fail	-	-	Yes	Yes
11	-	Fail	-	Yes	Yes
12	Fail	Fail	-	Yes	Yes
13	-	-	Fail	Yes	Yes
14	Fail	-	Fail	Yes	No
15	-	Fail	Fail	Yes	No
16	Fail	Fail	Fail	No	No

Table 2. SAM - scout, other-1 and other-2 - attackers.

In experiments 1 and 9 no failure occurred (a "-" in columns 2-4), and SAM's capabilities were not applicable. There are thus 14 cases where at least one agent fails. Of those, SAM detected a failure in 11, and generated a diagnosis for 9.

Experiment 2 is the original failure scenario. Here, as mentioned in the previous section, SAM detected a difference between its own "fly-flight-plan" and the other agents' "wait-at-point" and through diagnosis, realized the other agents believe the landmark is detected. It then recovered from the failure by adopting this belief, compensating for its own sensors' failure to detect the landmark. Experiment 13 is analogous, except that here SAM is playing the scout. SAM proves successful there as well.

There are 6 cases (experiments 3, 5, 7, 10-12) where a reverse difference occurs, in which the agent running SAM has detected the landmark successfully, but one or more other agents did not (and are continuing with "fly-flight-plan"). In all but one of these, a diagnosis is completed and recovery facilitated. This is a particularly strong result, since here a single agent deploying SAM is detecting a failure in one or more of its peers' perceptions.

In cases where two out of the three agents fail, SAM does better in detecting failures when working as the scout (capturing all of the cases 12,14,15), than when working as the attacker (not capturing all of 4,6,7). This is due to the fact that as a scout, even when SAM was failing, usually one of the two attackers would unambiguously signify detecting it by landing. In contrast, in case 4, when SAM was failing as an attacker, the scout is the only agent that did not fail. Here, since the scout continues to fly out to the battle position, and since SAM has not seen the landmark, SAM's plan-recognition fails to detect the scout's internal operator shift to "wait-at-point". Instead, SAM believes that the scout is continuing with "fly-flight-plan", and so no operator difference is generated. This shows a limitation of relying exclusively on plan-recognition in SAM, since plan-recognition may sometimes fail. Indeed, cases 14 and 15 further illustrate this point. These cases are also difficult to diagnose given the ambiguity in plan-recognition. Here, when one attacker detected the landmark and landed, while the other continued to fly along, the team was essentially split, and SAM's plan-recognition failed to unambiguously infer whether the team as a whole was running "fly-flight-plan" or "wait-at-point". It could only identify that each of the other agents was running a different operator, and thus only detected the failure (but did not diagnose it).

Although SAM is sensitive to plan-recognition ambiguity, this difficulty is alleviated since it focuses on teams and team-members. This constrains very much the space of possible hypotheses, and reduces the chance for ambiguity. Indeed, as the results of the experiments clearly show, there was only one case of detection failure and only two cases of diagnosis failure due to ambiguity.

SAM's two other failures at detection are both in cases where all three agents have failed together - which is another limitation of SAM. When all agents fail together, no differences in observed behavior or inferred operators occur, and so no failure is detected.

The combined results of the two sets of experiments demonstrate that just two agents running SAM--the scout and an attacker--are sufficient to detect failure, fully diagnose, and recover in *all 12 cases* where at least one agent behaves correctly (i.e., all cases except for those where all agents fail together).

Although we deliberately point out SAM's limitations and strengths as they show in the 14 experiments, its performance should also be compared to that of the techniques focusing only on single-agent perspective. Those would fail to even detect a failure in all 14 cases, even with all three agents running a diagnosis system. This is due to their lack of social attention to other agents. They consider only the agent's own sensors that are supposed to detect the landmark, but ignore the sensors telling them about what other agents are doing. Thus, they cannot diagnose or recover from failures in other agents.

A reviewer of this paper suggested contrasting these results with an evaluation of SAM running with no condition monitoring in place. In this case, none of the agents can detect the landmark, and so SAM will fail to detect the problem. This is to be expected, as SAM is a complementary technique, and does not consider the task-specific sensors that the condition monitors do.

SAM's results should also be contrasted with those achieved with imitation (Bakker and Kuniyoshi 1996). Imitation is a very special case of the general SAM method -- by choosing to always adapt the others' view, SAM leads to imitation. However, imitation works only in the presence of a *correct* role-model. The scout has a unique role in the team, and so has no role-model to imitate. Thus, imitation would be inadequate in all experiments in table 2. The attacker has a role model (the other attacker) and would manage to land correctly in experiments 2 and 6, but would also imitate its role-model when the role-model is failing (experiments 3 and 7).

Finally, to evaluate SAM's plan-recognition efficiency provided by model sharing, we compare the number of operators selected during a typical mission by an agent running with two other full hierarchies (as in a team with three agents), to that of the same agent doing modeling but using model-sharing for efficiency. The agent not using model-sharing keeps track of 57 different operators during the course of a typical mission. The latter agent, running with model sharing, keeps track of only 34 different operators--a two-fold computational savings.

The results demonstrate that SAM is a useful technique for detecting and diagnosing failures, able to capture failures in the agent running SAM and in its teammates, facilitating recovery not only of an individual agent, but of the team as a whole.

Related Work

We have already discussed some key related work—imitation and previous work on monitoring and diagnosis—earlier in the paper. In addition, SAM is related to work on multi-agent coordination and teamwork, although it generalizes to also detect failures in execution of individual operators, which are outside the scope of coordination. Particularly relevant are *observation-based* methods, which use plan recognition rather than communications for coordination. Huber and Durfee (1996) do not assume an explicit model of teamwork, but rather view collaboration as emergent from opportunistic agents, which coordinate with others when it suits their individual goals. These agents do not have the guarantees of maximal social similarity at the team level, and while they possibly will find the detected differences useful, they cannot be certain of failures, nor facilitate team recovery (since the other agent may simply have left the team opportunistically). Work on teamwork (Jennings 1995, Tambe 1997) concentrates on maintaining identical joint goals to prevent miscoordination, while the focus of SAM is on detecting when the goals do differ. Indeed. SAM is useful as a diagnosis component for general teamwork models, allowing a general teamwork replanner to take over the recovery process.

Atkins et al. (Atkins et al. 1997) attack a similar problem of detecting states for which the agent does not have a plan ready. They offer a classification of these states, and provide planning algorithms that build tests for these states. However, their approach considers only the individual agents and not teams. It thus suffers from the same limitations as other single agent approaches in being unable to detect modeled states which have not been sensed correctly.

Summary and Future Work

This paper presents SAM, a novel system for failure detection, diagnosis and recovery in large-scale, dynamic, multi-agent domains. SAM is not only able to detect failures in the individual agent deploying it, but also in that agent's peers--even if they are not deploying SAM themselves. SAM additionally diagnoses and recovers from failures at the team level -- it thus truly performs social diagnosis, spanning failures distributed in a number of agents rather than in one particular individual.

Key novel aspects of SAM include: (a) a new failure detection method, able to detect previously undetectable failures in the monitoring agent, and in its peers, (b) a social diagnosis method, able to diagnose failures in behavior of agents and agent-teams, based on an explicit teamwork model, and (c) a model-sharing technique for limiting the inefficiencies inherently associated with keeping track of multiple agent models. Finally, SAM exploits a novel synergy among three different agent components, specifically plan-recognition, teamwork, and monitoring components. These general principles would appear to be applicable in many domains such as those mentioned in the introduction.

In our work, we have implemented SAM in a complex, realistic multi-agent environment and conducted a careful and systematic evaluation of its benefits and limitations, demonstrating SAM's applicability in different single- and multiple-failure scenarios. We demonstrated that SAM was able to diagnose almost all failures in these cases, but approaches focusing on single-agent perspective have failed to even detect the failures.

Many areas are open for future work. Further exploration of Social Comparison Theory or the uses of model-based teamwork diagnosis is high on our agenda. We also plan to investigate further individual-operator differences, which do not enjoy the guarantees and diagnostic power of team-level failures.

Acknowledgments

This research was supported in part by NSF Grant IRI-9711665, in part by subcontract 090881-96-06 from Sverdrup Technologies, and in part by AFOSR contract #F49620-97-1-0501. Special thanks to K. Ushi.

References

Atkins, E. M.; Durfee, E. H.; and Shin, K. G. 1997. Detecting and reacting to unplanned-for world states, in *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*. pp. 571-576.

Bakker, P.; and Kuniyoshi, Y. 1996. Robot see, robot do: An overview of robot imitation. *AISB Workshop on Learning in Robots and Animals*, Brighton, UK.

Doyle R. J., Atkinson D. J., Doshi R. S., Generating perception requests and expectations to verify the execution of plans, in *Proceedings of AAAI-86*.

Festinger, L. 1954. A theory of social comparison processes. *Human Relations*, 7, pp. 117-140.

Firby, J. 1987. An investigation into reactive planning in complex domains. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-87).*

Grosz, B.; and Kraus, S. 1996. Collaborative Plans for Complex Group Actions. In *Artificial Intelligence*. Vol. 86, pp. 269-358.

Halpern, J. Y. and Moses, Y. 1990. Knowledge and Common Knowledge in a Distributed Environment., in *Distributed Computing* 37(3), pp. 549-587.

Huber, M. J.; and Durfee, E. H. 1996. An Initial Assessment of Plan-Recognition-Based Coordination for Multi-Agent Teams. In *Proceedings of the Second International Conference on Multi-Agent Systems.*.

Jennings, N. 1995. Controlling Cooperative Problem Solving in Industrial Multi-Agent System Using Joint Intentions. *Artificial Intelligence*. Vol. 75 pp. 195-240.

Kitano, H; Asada, M.; Kuniyoshi, Y.; Noda, I.; and Osawa, E. 1995. RoboCup: The Robot World Cup Initiative. In *Proceedings of IJCAI-95 Workshop on Entertainment and AI/Alife*.

Levesque, H. J.; Cohen, P. R.; Nunes, J. 1990. On acting together, in *Proceedings of the National Conference on Artificial Intelligence (AAAI-1990)*, Menlo Park, California, AAAI Press.

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

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

Tambe, M. 1996. Tracking Dynamic Team Activity, in *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96).*

Tambe, M. 1997. Towards Flexible Teamwork, in *Journal of Artificial Intelligence Research*, Vol. 7. pp. 83-124.

Toyama, K.; and Hager, G. D. 1997. If at First You Don't Succeed..., in *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97).* pp. 3-9.

Williams, B. C.; and Nayak, P. P. 1996. A Model-Based Approach to Reactive Self-Configuring Systems. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96).*