Implementing Shared Mental Models for Collaborative Teamwork

John Yen, Xiaocong Fan, Shuang Sun, Rui Wang, Cong Chen, Kaivan Kamali
School of Information Sciences and Technology
The Pennsylvania State University
University Park, PA 16802
jyen@ist.psu.edu, zfan@ist.psu.edu

Richard A. Volz
Department of Computer Science
Texas A&M University
College Station, TX 77843
volz@cs.tamu.edu

Abstract

Psychology studies have shown that one of the keys to human teamwork is the ability of teammates to anticipate the needs of others and proactively take appropriate action using an overlapping shared mental model (SMM). This paper introduces an implemented multi-agent architecture – CAST, which enables a team of agents to establish a computational shared mental model. Using such a SMM, agents can (1) anticipate others' information needs, (2) perform dynamic task allocations with minimum inter-agent negotiation, (3) choose whether to proactively inform teammates about the information relevant to their needs, and (4) rationally fuse information to meet teammates' different levels of information needs. We evaluate our approach through two experiments in simulated battlefield domains. The studies show the vision of empowering agents with SMMs for proactive teamwork behaviors is both challenging and promising, which can benefit the development of agent-based solutions for training and supporting mixed human-agent teams that need to filter and fuse an overwhelming amount of information, and to make critical decisions under time constraints.

1 Introduction

Shared mental models are thought to be an important aspect of teamwork among human teams. The notion of shared mental models is a hypothetical construct that has been put forward to explain certain coordinated behaviors of teams, based on a number of studies by cognitive psychologists [4, 5]. Basically, a shared mental model represents each team member's model of the global team state. This representation produces a mutual awareness, with which team members can reason not only about their own situation, but the status and activities of the other members of the team and progress of the team toward its goal.

The scope of a shared mental model is rather broad. Most collaborating agents have *shared goals*, either explicitly or implicitly. In traditional client/server distributed systems, the clients implicitly have a shared understanding about the server's goal to provide certain services to the clients. Because these goals do not change, they are embedded in the codes of the server. Software agents, on the other hand, often have their shared goals explicitly represented, so that they can adapt their goals to a dynamic environment. Joint intention theory formally defines the meaning and implications for multiple agents to commit to a shared goal (i.e., joint intention) [6]. The theory requires a team of agents with a joint intention to not only each try to do its part in achieving the shared goal, but also commit to informing others when the agent detects that the goal has been accomplished, becomes impossible to achieve, or becomes irrelevant. From the viewpoint of SMM, this means that agents are committed to maintaining a shared mental model about the status of the shared goal.

A shared understanding about team structure (e.g., different roles in a team) enables an agent in the team to develop a higher level abstraction about capabilities, expertise, and responsibilities of other team members. Shared knowledge about the process of a team provides a roadmap/recipe about how the team plans to accomplish its goal. In military, for instance, such a team collaboration process of a particular echelon unit (e.g., a battalion) is specified in its

"operational order". A teamwork process specifies not only "what should be done", but also specifies constraints on "who should do what". *Shared team structure* and *shared team process* together are essential for an agent to reason about the allocation of tasks among team members, and to collaborate smoothly as a team. An approach related to shared team process is the SharedPlan theory [8, 9], which is a formal framework for a group of agents to synthesize their full shared plan from partial ones.

In addition, a SMM may also include shared ontology, shared domain knowledge, etc. These elements are typically implicit in human teams. For instance, effective human teams usually have implicit ontology in mind in their communication.

The vision of our research is to empower agents with computational shared mental models (SMM) for enhancing their proactive teamwork behaviors. This is desirable when studying teamwork issues in mixed human-agent teams, because it will provide a computational support for mutual modeling between human and software agents. On the other hand, the vision is also highly challenging due to the broad scope of shared mental models and the complexity in implementing this cognitive construct.

To tackle the challenges, we have focused on a component of SMM – shared team process (and related team structure). In doing this, we implemented a multi-agent architecture – CAST (Collaborative Agent architecture for Simulating Teamwork), which enables a team of agents to establish a computational shared mental model using knowledge about the structure and the process of the team. This computational representation of SMM plays a critical role in allowing CAST to find a desired balance between achieving efficiency and maintaining adaptability in teamwork modeling. Using such SMM, agents can (1) anticipate information needs of teammates, (2) perform dynamic task allocations with minimum inter-agent negotiation, (3) choose whether to proactively inform teammates about the information relevant to their needs, and (4) rationally fuse information to meet teammates' different levels of information needs.

This paper introduces the SMM implemented in CAST and discusses how it can be used in several aspects to achieve effective teamwork. The rest of this paper is organized as follows. In the next sub-section we give the background of studies regarding shared mental models in human teams. Section 2 introduces the key features of the CAST agent architecture, and section 3 focuses on the computational shared mental model implemented in CAST. We describe the composition of the SMM, and discuss how the SMM is used in making decisions on whether to inform others, and how the SMM is used in fusing information. These two uses of SMM are evaluated through simulated experiments in section 4. Ongoing attempts of using the SMM in mixed human-agent teams are briefly discussed in section 5 and section 6 concludes the paper.

1.1 Studies about Shared Mental Models

Shared mental models extend the notion of individual mental models to a team context. An individual mental model about individual activities empowers the agent with "know-how" under different situations. Likewise, a shared mental model about team process can force the consequences of a situation to be worked out and realized, and allow predictions to be made, which can be very useful for anticipating what is likely to happen in the future.

Shared mental models are the key to supporting many interactions within a team that lead to its effectiveness and efficiency. For example, although communication is important to teamwork, Orasanu [12] found the counter-intuitive result that the higher-performing teams actually communicate less in high-tempo (or high workload) scenarios, which was attributed to greater implicit coordination made possible through enhanced shared mental models. Similarly, shared mental models are needed for proactive helping behaviors [13], where team members with less load offer to help members over-loaded with more tasks than they can handle, which again relies significantly on maintaining mutual awareness. The point is that team members have to have a model of what each other is doing in order to effectively collaborate, find synergies, avoid interference, assess relevance of information, etc.

Due to the importance of shared mental models to teamwork, fostering the development of shared mental models has been the target of several team training efforts. Most work on team training begins with (or relies on) identifying a method for assessment. Shared mental models can be measured in terms of the degree of overlap or consistency among team members' knowledge and beliefs [5]. Methods to increase the overlap of dynamic information (e.g. about situation, workload, etc.) include encouraging frequent communication of individual updates during a scenario to keep other members of the team informed whenever inconsistencies or changes occur. A popular method aimed at increasing the overlap of static knowledge (e.g. of roles and responsibilities) is cross-training [1, 3, 16].

2 The CAST Agent Architecture

CAST (Collaborative Agent architecture for Simulating Teamwork) is a teamwork model that enables agents, whether they are software agents or human agents, to anticipate potential information needs among teammates, and to exchange information proactively [20]. It has been designed to study teamwork-related issues emerging from hierarchical teams with well-defined process, distributed expertise, and limited communication in time-stress domains.

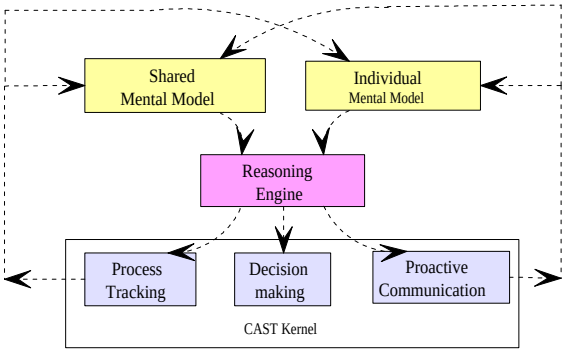


Figure 1. The CAST Agent Architecture

In the CAST agent architecture as shown in Fig.1, there are mainly five integrated components related to the Reasoning Engine: Shared Mental Model, Individual mental model, process tracking, proactive communication, and decision making. Briefly, the Reasoning Engine, based on the current shared and/or individual mental states, triggers appropriate algorithms in CAST kernel to monitor the progress of team activities, do dynamic agent assignment, anticipate others' information needs, make decisions on proactive communication actions, and fuse information to meet higher-level information needs, etc. The execution of these mental actions will reversely affect the evolution of the shared and individual mental states.

The *Shared Mental Model* component involves all the knowledge and information that are shared by team members, and all those of which the team members should commit to the shared awareness. We will discuss this component in the next section.

Contrary to SMM, *individual mental models* deal with those mental attitudes held by individual agents rather than all team members. It is continuously updated by sensor inputs and communication messages received from other agents. Theorem-prover JARE (Java Automated Reasoning Engine) [11] is used by each individual agent to manage beliefs about the dynamic world; test constraints, answer queries and make inferences using the belief base and domain-dependent Horn-clauses. An agent's individual mental model may also include its model of other's mental state, i.e., nested mental models. CAST supports belief reasoning regarding other teammates through reasoning about their observability, or inferring from the effects of the actions already completed, as long as the knowledge about observability and action effects are part of the shared mental model. This is especially important in enabling CAST agents to anticipate other teammates' information needs, and it highly impacts the information exchanges among team members. For instance, suppose agent A can observe an enemy if the enemy is within his sensing range (in general, to determine observability can be more complicated than only considering the distance to the target. Weather and terrain, for instance, may also affect observability). Given information about the location of A and an enemy, a teammate of A can infer whether A can observe the enemy based on the knowledge about A's sensing range. The teammate may choose to inform the enemy information to A proactively if he infers that A cannot observe it.

The *Process Tracking* module encapsulates two algorithms: *PrT-Interpreter* and *Dynamic-Agent-Assignment* (DAA). The former algorithm is used for interpreting (manipulating) the team process so that all the team members could coordinate their behavior to embody the specified team behavior. In CAST, team processes are originally coded in MALLET (Multi-Agent Logic-based Language for Encoding Teamwork), a logic-based language for specifying the structures and processes of agent teams [19]. To facilitate dynamic reasoning and monitoring, team processes are internally represented as PrT nets, which are generated offline by MALLET Parser. One of the tasks of PrT-Interpreter is to ensure all the team members behave, react, and deliberate strictly according to the committed (intended) plans (courses of actions), and synchronize their behaviors whenever necessary (i.e., maintain their shared mental model regarding the progress of team activity).

Initially, a team plan is only partially instantiated, though fully represented in a PrT net. The *DAA* algorithm is responsible for initiating appropriate communication actions at the appropriate time to evolve these partial plans, i.e., determine parameters such as the actual doers of certain actions. DAA first determines the current goal and finds an appropriate plan for it. Each partial plan specifies constraint conditions on how to allocate the embedded unresolved tasks. Such conditions may include role constraints, workload conditions, or domain related conditions. Before executing a task, each agent will check these constraints with its current beliefs about the domain or teammates and determine the suitable agents whose conditions satisfy the constraints. Every call of the DAA wrt. a certain plan will result in a "who-do-what" list, which will be further used to update the SMM regarding others' information needs.

One algorithm in the *proactive communication* module is called DIARG (Dynamic Inter-Agent Rule Generator), which is the most crucial part of CAST in terms of achieving the proactive assistance behavior in team environment. It is used to anticipate teammates' information needs and update information flow relationships dynamically. The foundation of DIARG algorithm has been established by extending the SharedPlan theory with the formal semantics of proactive communicative actions [17, 18]. It has been shown that an agent's consideration of proactive assist behaviors can be derived from axioms in the formal framework. A team's SMM about the information needs of its

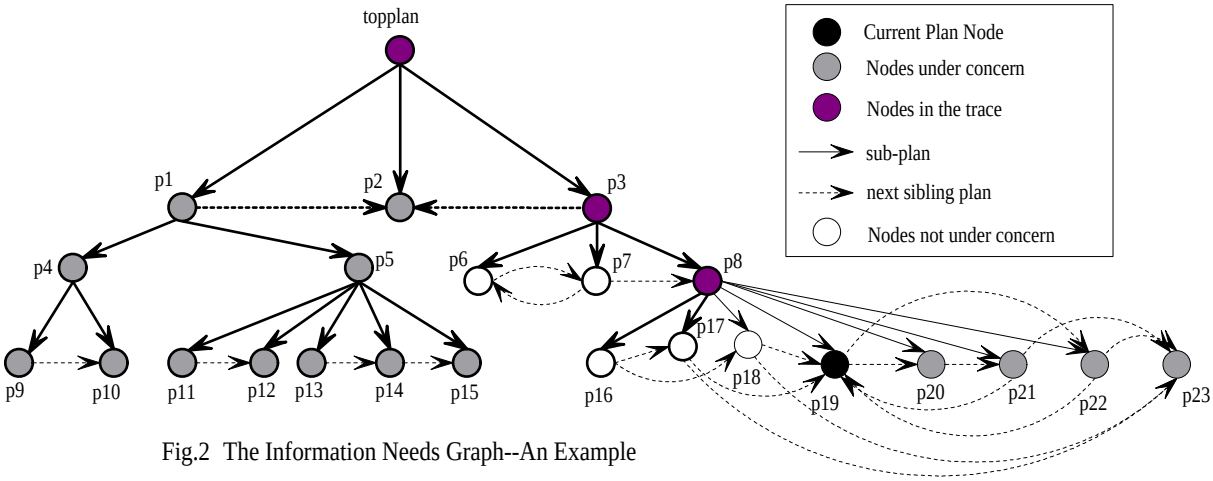


Fig.2 The Information Needs Graph--An Example

members is captured in the framework by a modal operator $InfoNeed(A, I, t, C_n)$, which says that agent A needs information I at time t under the context C_n. In CAST, this part of SMM is implemented as *Information Needs Graphs* (see Fig. 2 for an example), which can be manipulated dynamically to reflect the most updated information needs relationships among the team.

Information Needs Graphs (abbreviated as ING) are generated at compile time by analyzing the input team process specified in MALLET. In an ING, nodes are labeled with names of plans or primitive actions referred to directly or indirectly by the plan labeling the root node. There are two kinds of relations among the nodes. A *sub-relation* (e.g., between nodes $p1$ and $p4$) refers to the direct sub-plan relation in the process specification (e.g., plan $p4$ is directly referred to by plan $p1$). A *sibling-relation* (e.g., between nodes $p1$ and $p2$) refers to the implicit execution sequence embodied by the process specification (e.g., plan $p2$ will be executed next when plan $p1$ is done). Each node also records the potential doers of the plan and the pre-requisite information of performing the plan (the pre-requisite information can be structured in certain ways. See Section 3.2 for detail).

The INGs can be used to better anticipate others' information needs. For the example in Fig.2, $p19$ is the current plan node, which means the corresponding agents are executing plan $p19$. If the team activities are synchronized, the doers of $p19$ can assume all those information required for plans $p6$, $p7$, $p16$, $p17$, and $p18$ are no longer needed, since they have already been completed. Such dynamic anticipation can significantly reduce unnecessary information delivery as team activities proceed. Even though agents may not act in the same tempo, an agent may be able to predict (recognize) others' current plans based on their observable behaviors. Then, the INGs can still be leveraged to improve the accuracy in anticipating others' information needs.

The DIARG monitors information newly sensed by an agent to determine whether it matches the information needs of some teammates. If there is a match, by default the agent will send out the new information to the corresponding needers proactively. Such information delivery behavior could be more rational when considering the cost and benefits from the perspective of the whole team. To do so, this module needs to interact with the decision making module.

The decision making module currently consists of two algorithms enabling agents to: (1) decide on the course of actions (COA) by using designated decision maker, and (2) decide whether to send out information proactively ($ProInform$). The latter will be explained in Section 3.1.

3 The Computational SMM in CAST

Shared mental models typically consist of a wide array of types of information [15], some static and some dynamic. Static information includes knowledge about the structure of the team (who is playing what role), the nature of the roles (i.e. responsibilities), capabilities possessed by team members or required by roles, the team goals (e.g. missions), the selected plan for achieving a goal, communication policies, etc. Dynamic information includes more transient aspects, which might change throughout the course of the scenario, such as workload, current task assignments, status of tasks (pending, blocked, completed), and overall progress of the team process toward its goal.

As shown in Fig.3, the Shared Mental Model implemented in CAST has four components: team process, team structure, domain knowledge, and information needs graphs. The *team process* component can be further split into static part and dynamic part. The static part is a collection of plans represented as PrT nets, which describe how the team is to accomplish its goals. These plans are

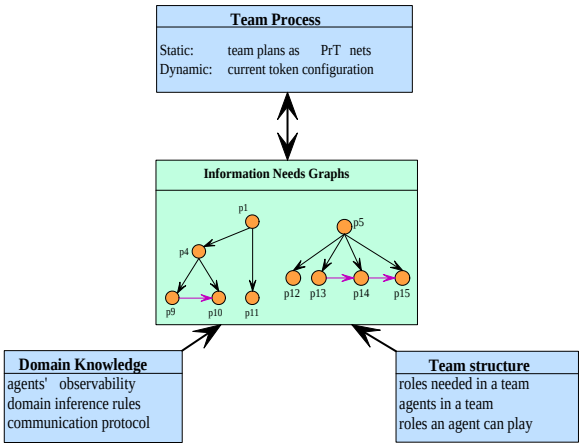


Fig. 3 The Composition of Shared Mental Model

more like incomplete recipes in the SharedPlan theory, since agent task allocation needs to be done dynamically before the plans can be executed collaboratively. The dynamic part is a collection of token configurations, each of which tracks the current progress of the corresponding plan. The *team structure* component captures those knowledge specifying roles in the team, agents in the team, and the roles each agent can play. The *domain knowledge* component describes domain-dependent static common knowledge shared by all the team members, such as each agent's observability (used to approximate nested beliefs), communication protocols, inference rules, domain expertise, etc. The *information needs graphs* component maintains the dynamic information needs relationships (i.e., make sure the information needs reflect the current status of team activities). The manipulating of information needs graphs requires the meta-information provided by the other three components.

The information needs in CAST can be classified into three categories. First, CAST is implemented such that each team member is committed to letting others know its progress in the current team process. Such communication for synchronization purpose is motivated by the *built-in* information needs: each agent needs to know others progress in order to maintain the SMM regarding team process (i.e., the dynamic part). These built-in information needs are crucial to achieve teamwork in CAST: on the one hand, they are part of the shared mental model; on the other hand, they are used to maintain the other part of the shared mental model. Second, by parsing the MALLET specification, CAST can construct incomplete information needs graphs from team processes at compile time. These information needs embedded in INGs are called *pre-computed* needs. Third, the pre-computed information needs can be further *refined* at execution time as the team do dynamic agent assignments.

The major difference between mutual beliefs (MB) and SMM is that the MB construct emphasizes recursive nested awareness of certain facts, while SMM emphasizes more on the *strong commitment* from the involved agents towards maintaining certain shared awareness. In other words, having a shared mental model, all the agents are committed to eliminating any differences that may emerge at any time. For instance, when an agent finishes its task, the agent will inform others that it will proceed to the next task. This communication is entailed by its commitment to maintaining the shared awareness of the progress of team activities. It is such commitments that force individual agents to communicate their private information, if necessary, to teammates, just like the role joint intention plays in the joint intention framework [6].

In addition to the *maintenance* of a shared mental model among the agents, another novelty of our architecture is how to *leverage* the computational shared mental model to achieve effective teamwork. As we mentioned, in DAA the SMM is used to do dynamic task allocations among teammates (this process also allows agents to update their SMM regarding responsibility of teammates), and in DIARG the SMM is used to anticipate others' information needs. In the next, we will discuss two other uses of SMM in detail: (1) how SMM can be used to decide whether to assist teammates regarding detected information needs of teammates when communication cost is taken into consideration, and (2) how SMM can be used to guide information fusion in order to meet others' higher-level information needs.

3.1 Deciding Proactive Communication Using SMM

As a kind of helping behavior, an agent will consider proactively inform ($ProInform$) other teammates when it believes delivering the information will enable the receiving agents to perform actions or make better decisions. However, when communication cost (risk) is considered, it's not necessary that proactive information delivery can always benefit the whole team. For instance, the messages to teammates may be overheard by agents in an opposite team, which may induce the opposite team to change its tactical strategy to attack the first team. Hence, rational agents should be able to evaluate the utility of $ProInform$ vs. the potential cost before actually doing it.

Generally speaking, factors that may affect the decision can be grouped into two categories: (1) communication costs to self, and (2) potential impacts of the decision to teammates. The first factor may involve uncertainty due to the characteristic of the environment. For instance, if an agent communicates in a battle space, it may be detected by the enemy with a certain probability (based on knowledge about the sensor capabilities of the enemy). The second factor is the agent's belief about how useful the information is

to the teammate, and how much damage the teammate will suffer if the information is not delivered.

Our decision-theoretic communication strategy considers the utility of the communication action, its cost, and the uncertainty of the different possible outcomes in calculating an expected utility of the $ProInform$ communicative action. A similar calculation can be made for the expected utility of not taking the action. Agents simply chooses the decision with the highest expected utility [10]. In other words, if the expected utility of the action exceeds the expected utility of not taking the action, a CAST agent, being a rationale one, will choose to proactively deliver the information to the teammate who needs it, even if such communication may introduce risk to the agent.

More formally, let a_i denote a possible decision regarding whether to proactively communicate a particular piece of information to a specific information needer (say, agent B), O_j denote the possible outcomes of that decision, $P(O_j|a_i)$ denote the conditional probability that outcome O_j will occur given that the agent chooses a_i, $C(O_j)$ denote the communication cost of the outcome O_j, and $U_B(O_j)$ denote the utility of outcome O_j to the information needer (B). The expected utility of the decision a_i is

$EU(a_i) = \sum_j P(O_j|a_i) \times (U_B(O_j) - C(O_j))$.

However, it's not easy to get the expected utility. To do this, agents need to know all the potential outcomes of the decision, and all the corresponding values of $C(O_j)$, $U_B(O_j)$, and $P(O_j|a_i)$. One difficulty is that all such factors are domain and situation dependent. For instance, communication cost may involve resource constraints (network bandwidth constraints, the energy required, etc.) and the potential risks due to communications. While the resource constraint is often known and static, communication risk typically depends on the situations (e.g., depending on the location of a scout agent, sending a message may or may not be detected by enemies). The impact of a communication decision on the receiving agent is even more complicated. This may require the decision maker to know the utility of a piece of information to the potential receiver, which is even harder to figure out. For example, suppose an attack mission requires two bombers to strike the enemy target simultaneously. Informing threats from an approaching enemy aircraft may protect a bomber from being destroyed. However, the utility of informing threats depends on how many bombers are alive (and possibly their distance to the target).

We deal with this challenge through collecting the abovementioned domain-related knowledge either by doing large-scale experiments (e.g., the probabilities) or from domain experts, and encoding the expertise as part of the Shared Mental Model maintained by all the agents in a team. Such expertise can then be used (e.g., through case-based reasoning) in calculating the expected utilities of communication decisions.

In deciding whether or not to deliver newly sensed information to the needers, the Shared Mental Model can also be used in another way. An agent's belief about whether the potential receiver already knows about the information also affects its decision making. This kind of beliefs can be approximately established based on an agent's SMM about the teammate's observability.

3.2 Fusing Information Using SMM

Level two processing of the JDL Data Fusion Process Model [2] uses the raw data combined and refined at level one processing to develop a description of relationships among entities and to interpret the current situation. For example, level 2 fusion can be used to develop an interpretation of the composition and disposition of the local threat forces and their current activities [14]. Here we show how the SMM with regard to the structure of information needs can be used in this means.

As mentioned, information needs graphs enable agents to make better anticipations regarding others' information needs by tracking the progress of team activities. Each node in an ING stands for one plan or primitive operator (for leaf node) called directly or indirectly by the plan labeling the root node. The potential doers and the preconditions of a plan are also recorded in the corresponding node of an ING.

Another key feature of CAST is that the preconditions of each plan are constructed as a tree structure. That is, for each node in an ING (as shown in Fig. 2) there is an embedded tree capturing the different abstract levels of preconditions. Fig.4 shows an example of such precondition trees, where $r1, \cdots, r10$ are predicates describing certain domain related information required to executing the corresponding plan. In the construction time of an ING, when a node corresponding to some plan p_i is created, the precondition tree for p_i can be generated as follows. If the precondition of p_i is a single predicate, create a root node labelled with the predicate; if the precondition is composed of a set of predicates, create a virtual root node, then create a node for each of the predicates and make these nodes sons of the virtual root node. Then, populate the tree by applying the following recursive algorithm to the leaf nodes.

Algorithm: populateTree(Node nd, Predicate pd)
```
  /* get the Horn Clause with pd as its head */
  hc = getHornClause(pd);
  if (hc is Null) return;
  /* get all the negative literals of hc */
  plist = tail(hc);
  for each pred in plist
     pn = createNode(pred);
     addSon(nd, pn);
     populateTree(pn, pred);
end.
```

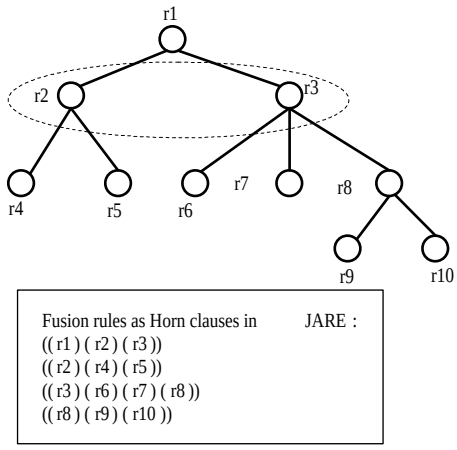


Fig. 4 A Precondition Tree

The precondition trees can be used in at least two ways. First, they can be used in collaborative constraints satisfaction. Suppose we have the precondition tree as shown in Fig. 4 for some plan p_i, where the root is a virtual node (i.e., the preconditions of p_i is specified as $(r2\ r3)$). Note that in a precondition tree, the nodes at the same level collectively form a context for each individual node. For instance, in this example, $r2$ is useful only when it is evaluated together with $r3$, thus, $(r2\ r3)$ establishes a context for both $r2$ and $r3$. Now, suppose agents A_1, A_2 and A_3 share this precondition tree, and A_3 is the doer of plan p_i. Also suppose A_1 has (no) the information described by $r2$ ($r3$), and A_2 has (no) the information described by $r3$ ($r2$). In this case, obviously neither A_1 nor A_2 alone can enable A_3 to do p_i. However, they can collaboratively satisfy A_3, because A_1 knows $r2$ will be useful for A_3 in the context $(r2\ r3)$ and A_2 knows $r3$ will be useful for A_3 in the context $(r2\ r3)$. Here, for A_1 and A_2, the direct information needs of A_3 is $(r2\ r3)$ as a whole, but both A_1 and A_2 can infer the indirect information needs (i.e., $r2$ and $r3$ in separate) of A_3 by reasoning on the shared precondition tree.

Second, the precondition trees can be used to guide information fusion by breath-first reasoning. For the example shown in Fig.4, suppose the predicate $r1$ labeling the root is the precondition specified for plan p_i, agent A_1 can observe the information described by $r2$ and $r3$, and agent A_2 is the doer of plan p_i. Assume that what A_2 needs (i.e., $r1$) can not be directly observed from the world. To meet A_2's needs, A_1 can query its reasoning engine to check whether $r1$ holds or not. According to the Horn clause $((r1)\ (r2)\ (r3))$ (i.e. $r1 \leftarrow r2, r3$), the truth value of $r1$ depends on the truth values of $r2$ and $r3$. Being aware of $r2$ and $r3$, A_1 can fuse $r2$ and $r3$ to create $r1$ and inform $r1$ to A_2 proactively. Alternatively, knowing $r2$ and $r3$ are necessary in deriving $r1$, and knowing the inference knowledge (i.e., the relevant Horn clauses) is also shared by A_2, A_1 may choose to send $r2$ and $r3$, while leaving the task of fusion computing to A_2 itself. In such case, for A_1, information $r2$ and $r3$ are indirectly needed by A_2. The difference of the two alternatives is who will do fusion computing, the information provider or the information consumer.

Information fusion can also be carried out by depth-first reasoning. One difference is that breath-first reasoning guarantees that the higher-level information will always be delivered with higher priorities than the lower-level information, which ensures the information consumers consider higher-level information first. As we will discuss later, This is more significant when the the information consumer only has limited cognitive capacity.

4 Experiments

To evaluate the impacts of the SMM implemented in CAST on team performance, we conducted two simulation experiments. The objective of the first experiment is to evaluate how CAST's decision-theoretic approach to proactive communications affects the performance of the team, while the objective of the second experiment is to evaluate how information fusion may improve team performance in terms of tactical decision makings when the decision-maker's cognitive capacity is under consideration. Both experiments are carried out in a simulated battlefield represented by a 21×21 grid world, where two opposing agent teams, a friendly team (blue) and an enemy team (red), navigating in the combat field to achieve certain team goals.

4.1 Experiment I: The Decision Theoretic Communication Strategy

4.1.1 Scenario and Experiment Design

In the first experiment, the blue team adopts an offensive strategy: the team goal is to destroy the homebase of the red team. The red team tries to protect their base by attacking any approaching agents of the blue team. The blue team is formed by agents with three different roles: the scout, who can sense but can not shoot; the fighter, who can shoot but can not sense; and the bomber, who can only bomb the enemy base. To complete the mission, at least two bombers have to surround the enemy base and perform a joint action called co-fire at the enemy base. The structure of the blue team is designed based on four assumptions: (1) members of the team have limited and different observability, (2) the team needs to act under critical time constraints, (3) communication cost is not negligible, (4) the team has a well-defined structure and process. For instance, agents with different roles in the blue team have different observability. Furthermore, each agent's observability is limited. The

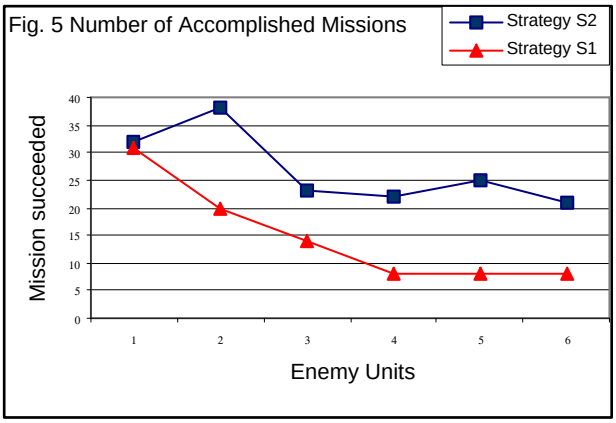


Fig. 5 Number of Accomplished Missions

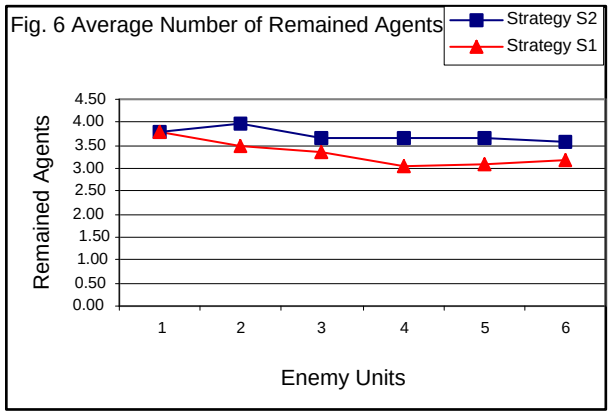


Fig. 6 Average Number of Remained Agents

scout agent can only sense enemy agents that are within its sensing range. To ensure that agents can coordinate according to its plan, the scenario requires that at least two bomber agents synchronize and bomb the enemy base at the same time in order to accomplish the mission.

The behavior of the blue team is governed by team plans and individual plans specified in MALLET, along with other related domain knowledge. The team will coordinate their movement toward a series of waypoints which ultimately leads to the enemy base. Being informed about the location of the enemy base, the bombers will move toward the enemy base and try to synchronize co-fire actions to complete the mission, while the unassigned fighters will also move toward the enemy base to protect bombers whenever needed. When informed about the location of a moving (red team) enemy, a dynamically assigned fighter (based on the team's SMM about the constraint of the assignment and workload) will move toward the enemy's location and shoot at it. Meanwhile, if a bomber gets such information it will try to move away from the threat.

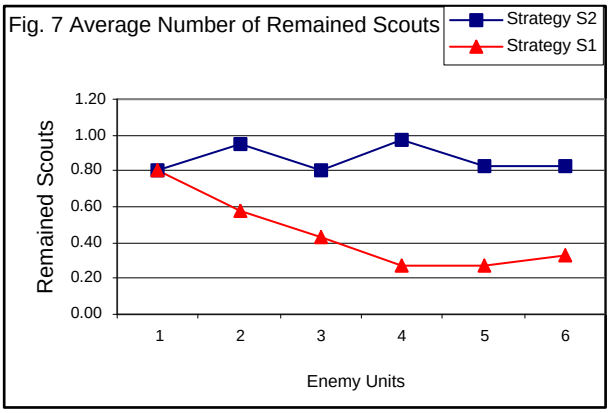
The red team involves three types of agents: one agent protecting a waypoint on the blue team's attack plan, one agent protecting the homebase, and other agents patrol around the homebase following a circle whose radius is determined by their initial locations, which are randomly assigned. Unlike the blue team, the red team agents do not have a shared mental model. They each act independently without communicating with each other. The only coordination in the red team is that the enemy base sends the location of detected intruders (i.e., blue agents) to a red team agent who is closest to the intruder (for initial assignment) or to a red team agent who has been previously assigned to the intruder (for tracking later).

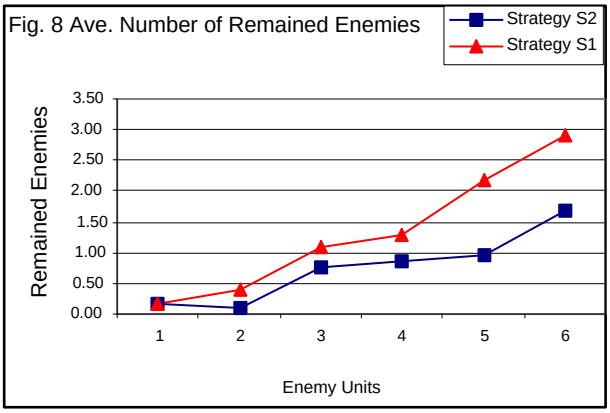
Each agent in the red team has a sensing range. Even though this sensing range is slightly shorter than that of the scout agent in the blue team, the range is much further than that of the fighters and bombers in the blue team. When a red agent detects a blue agent, the red agent will move toward the blue agent, shoot at it and kill it once the blue agent is within the shooting range.

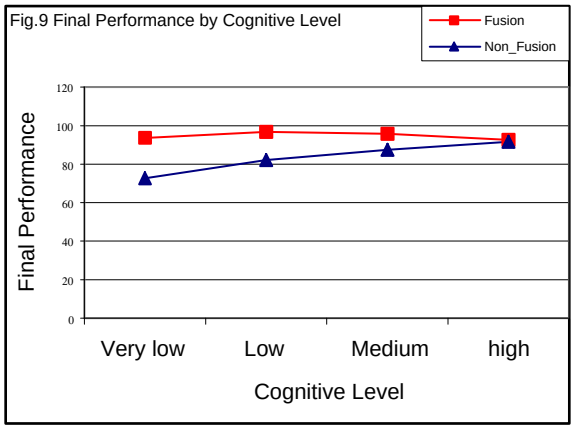
To ensure that the communication cost is not negligible, we designed the scenario to introduce a risk factor into an agent's communication: a risk that a blue agent's location can be detected by the enemy base if the agent sends a message within a "detection ring" of the enemy base. If the red agent base detects the location of the blue agent, it informs one of the enemy agents about it so that it moves toward the location and tries to shoot at it.

In this experiment, we designed two communication strategies for the blue team to study the effectiveness of our algorithm for decision-theoretic evaluation. The two strategies differ on how they handle the communication decisions on whether to proactively inform bombers and fighters about enemy agents' location. The strategy $S1$ always informs the closest bomber about detected enemies so that the bomber can escape. $S1$ also always informs fighters about enemy locations so that the fighters can move toward the enemies and attack them. $S2$ adopts a decision-theoretic approach to choosing whether to inform teammates about the detected enemies. If the expected utility for proactive communication exceeds the expected utility of not communicating, the agent will choose to communicate. Otherwise, it will withhold the communication. It is worth noting that when the decision of $S2$ is "ProInform", the effect is equivalent to the effect of using $S1$, whether the decision is right or not; while when the decision of $S2$ is "not ProInform", the effect is better than that of $S1$ if the decision is right, and the effect is worse than that of $S1$ if the decision is wrong.

In addition to communication strategies, we also use number of enemy units, ranging from 1 to 6, as another independent variable, which indicates the level of domain complexity. Obviously, the more enemy units protecting the waypoint and the homebase, the more difficult the mission becomes. For the red team, when the configurations have two enemy units, they patrol around the enemy base and







the waypoint respectively; when configurations have only one enemy unit, it simply patrols around the enemy base. For each configuration, we generated 40 missions by randomly assigning initial positions of the agents. Blue team agents were randomly assigned within the upper left 5×5 "launching region" of the mission. The enemy unit agents were randomly assigned to a location within a radius of the target (e.g., the enemy base or a waypoint) it is protecting.

4.1.2 Result Analysis

Fig. 5 summarizes the number of successfully completed missions (out of 40 missions in total) for the blue team using the two communication strategies. As shown in the figure, strategy $S2$ outperformed strategy $S1$. The performance difference between the strategies was more significant when the number of enemy units increased. These experimental results suggest that the decision-theoretic communication strategies can be effective for team-based agents to decide on whether to proactively deliver needed information to teammates when communication carries a cost.

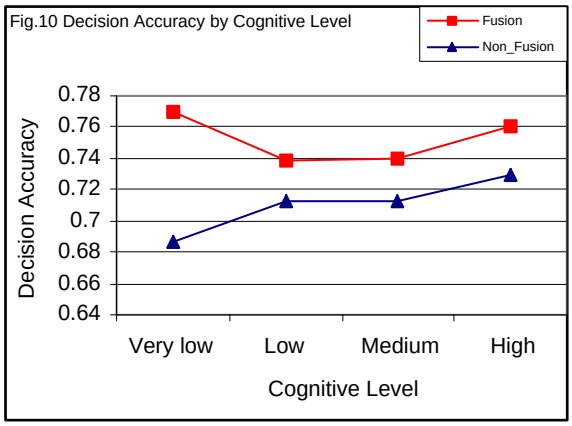
To gain a better insight about the results, Fig. 6, 7, and 8 show the average number of blue team agents, blue team scout agents, and enemy unit agents that survived. While the two strategies resulted in slight difference on the number of survived blue team agents, they differ significantly on the scout agent. In fact, the decision-theoretic communication strategy ($S2$) was able to maintain the survival rate of scout as the number of enemy units increased, as shown in Fig. 7. In contrast, the survival rate of scout agent decreased rapidly under the alternative communication strategy ($S1$). These observations suggest that considering the risk factor of agent communications help to protect the scout agent, which contributes to higher mission success rate.

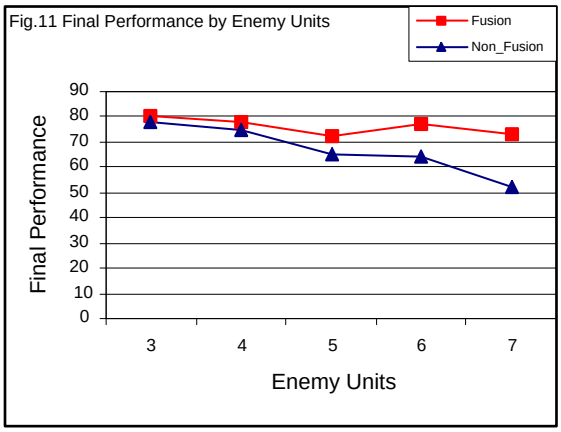
4.2 Experiment II: The Effect of Information Fusion on Decision Making Tasks

4.2.1 Scenario and Experiment Design

The purpose of this experiment is to understand how significant the information fusion is to team performance, especially to the accuracy of decision-making tasks, when the information consumer has only limited cognitive (information processing) capacity. The long-term goal of this study is to explore solutions to the dilemma emerging in BattleSpace InfoSphere: the more information the better tactical decisions can be made, but not too much when the cognitive capacity of the decision maker is considered.

In this experiment, the blue team adopts a defensive strategy: the blue force is trying to prevent assault from the red force. Collaborating for the blue side are three battle functional areas (BFAs): the intelligence cell ($S2$), the operations cell ($S3$), and the logistics cell ($S4$). For the purpose of this scenario the goals of the BFAs have been simplified and defined as follows: $S2$ has as its objective the assessment of all enemy locations, actions and intent; $S3$ has as its objective the defeat of the enemy and protection of the supply routes; and $S4$ has as its objective the identification of supply routes and sustainment of supplies. We use CAST agents to play the role of $S2$, $S3$, $S4$, fighting forces under





the control of $S3$, and unmanned-aerial-vehicles (UAV) under the control of $S2$.

To make a comparison, two blue teams are designed: a fusion team and a non-fusion team. In the fusion team, $S2$ fuses the information collected by UAVs and other BFAs into higher-level information, and send it to $S3$ for making decisions. In the non-fusion team, $S2$ simply forwards the relevant information collected by UAVs to $S3$, which has to take some effort in processing the lower-level information before making decisions. At each time steps, based on the information currently available, $S3$ tries to make decisions on whether to fire at the approaching enemies depending on whether the enemies introduce a high-threat to the main supply route. To simulate the behavior of logistics, $S4$ is designed such that it moves along the main supply route, and changes the supply route whenever a threat comes. The game is over as soon as $S4$ arrives at the pre-specified target.

The members of the red team are randomly launched and moving nearby the main supply route to introduce potential threats to blue teams. For each run, the number of enemy units and their routes are randomly configured.

For the first set of experiments, we manipulate the cognitive capacity of $S3$ for both blue teams to examine its effects on final team performance and the decision accuracy. For simplicity, we simulate the cognitive capacity as a limited memory queue. Being overloaded means the agent cannot handle the information beyond its limited memory queue. We distinguish four levels of capacity ranging from "very low", "low", "medium" to "high".

We launched 400 runs for each blue team. Each point in Fig. 9 and Fig. 10 corresponds to the accumulated results out of 100 runs (20 runs for each configuration of enemy units ranging from 3 to 7). Fig. 9 shows the number of missions succeeded, and Fig. 10 shows the decision accuracy computed by *num(correct-decisions)/num(total-decisions)*. $S3$'s decision is correct if it is the same as that computed by world simulator (which has complete information).

For the second set of experiments, along the dimension of domain complexity, the number of enemy units is varied. We launched 400 runs for each blue team using the number of enemy units as control variable (from 3 to 7). Each point in Fig. 11 and Fig. 12 corresponds to the accumulated results out of 80 runs. Fig. 11 shows the number of successful missions, and Fig. 12 shows the decision accuracy.

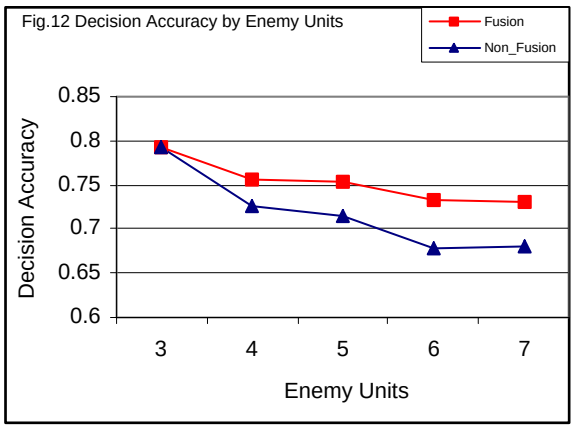
4.2.2 Result Analysis

According to Fig. 9, the fusion team in general outperformed the non-fusion team. Additionally, the performance difference between the fusion team and non-fusion team increased as S3's cognitive capacity decreased. From these results we may conclude that information fusion capability can be a very important factor of team performance when the information consumer has poor cognitive capacity. The same results are true when the decision accuracy is compared, as shown in Fig.10. Further, in Fig. 10 the decision accuracy of the non-fusion team increased as S3's cognitive level increased, while there is no such trend for the fusion team. This may suggest the decision accuracy of the non-fusion team is more sensitive to cognitive level changes.

When the number of enemy units acts as control variable, the fusion team also outperformed the non-fusion team in general, as shown in Fig. 11 and Fig.12. The performance and decision accuracy of both teams decreased as domain complexity increased. However, the gap in team performance and decision accuracy between the two teams became bigger as the number of enemy units increased. These results may indicate that the benefits of fusing information become stronger as the domain complexity goes up.

5 Other Attempts Using CAST

The CAST architecture empowered with computational shared mental model has being used in two other areas.



First, computational shared mental models are crucial for modeling team behaviors in the support of distributed team training. Applied to the team training domain, a SMM can be used to model the behavior or assess the performance of individual team members, sub-teams or the entire team, to provide intelligent coaching feedbacks and customize training sessions in term of human trainee's specific needs.

We are interested in implementing user modeling and intelligent coaching agents to teach trainees regarding collaboration skills. For example, knowing when to communicate and what information needs to be delivered is among one of the desired collaborative skills for trainees to learn. In terms of collaboration behavior, several characteristics of effective human team is considered, such as anticipating others' information needs and backing up overloaded teammates.

A user modeling approach has being developed leveraging the Shared Mental Model implemented in CAST, where human trainees can act as virtual agents interacting with team members including expert coaching agents. CAST *coaching agent* has three additional components– User Model (UM), Expert Model (EM), and Team Assessment Module (TAM). The *User Model* Component refers to trainer's model of a trainee, e.g., knowledge about the trainee's knowledge or lack of knowledge, its specific skills, etc. *Expert Model* component serves as a domain expert who knows what actions, especially collaborative actions, a trainee should do. The embedded SMM enables the expert model to automatically anticipate such needs for collaborations. The *Team Assessment Module* takes both inputs from human trainee (based on the monitoring results of user actions through user interface) and from the Expert Model. By comparing the expected actions from EM and the observed behavior of the trainee, the trainee's performance can be evaluated regarding the team tasks, and certain adaptive coaching feedbacks will be sent to the UM if necessary.

Second, CAST agents have being used in an extended Robocup Rescue domain where the center agents are played by humans. Due to the dynamic, complex nature of the disaster domain, agents have to collaborate with each other effectively to save civilians and extinguish fires under time pressure. A SMM serves as the support for agents to anticipate the needs of their teammates regarding disaster information and proactively deliver the information to them.

In this domain, center agents are responsible for making critical decisions regarding task allocations due to their relatively broader situation awareness compared with platoon agents. We choose humans to play the center agents because human's powerful spatial reasoning ability can be better leveraged in the decision making process.

To bring humans in the loop, each human has an associated interface agent, through which the human collaborates with other teammates. Both the interface agents and the platoon agents (ambulances, police forces and fire brigades) are customized CAST agents, so that they can maintain a computational SMM.

Based on the SMM maintained between the platoon agents and interface agents (simply model human's part), a platoon agent can proactively meet center agents' information needs (for making better decisions on task allocation) by sending the relevant information it collects. On the other hand, having a SMM with platoon agents, center agents can provide decision supports for them.

6 Conclusion

This paper focused on the cognitive construct "shared mental model", and presented our work on computational shared mental model and its use in anticipating teammates' information needs, in deciding whether to proactively send a piece of information, and in fusing lower-level information to meet teammates' high-level information needs.

The computational shared mental model is based on an extension to the standard PrT nets. Petri Nets have been tried before for simulating teamwork [7], and were found to be slow and difficult in maintaining. Our approach, however, has been shown to be much more successful. By separating domain independent aspects from domain dependent ones, teamwork models can be captured into several abstract levels, which allows an agent to track team processes in PrT nets only at the needed level of detail. In addition, by simulation experiments, we examined how shared mental model enables CAST agents to adopt proactive communication only when necessary, and how shared mental model can be used to fuse information to alleviate the cognitive overload problems faced by human decision makers.

The vision of our research – empowering agents with shared mental models for enhancing their proactive teamwork behaviors – is powerful, yet challenging. It is powerful because we can easily recall our personal experiences in which we benefit from proactive help from teammates,

colleagues, friends, and even family members. These experiences help us, and to some degree convince us, to imagine all kinds of agent-based services that may be possible under this vision. However, the vision is also highly challenging. The scope of "shared mental model" is very broad, and we need to achieve two conflicting goals regarding teamwork modeling using SMMs: high efficiency and high adaptability. Maximizing adaptability often introduces overhead for communication and coordination, while maximizing efficiency tends to reduce the adaptability of the team. Therefore, finding an acceptable tradeoff between these two conflicting goals has been a continuous challenge.

To tackle the challenge regarding the broad scope of SMMs, we have focused on a component of SMMs – shared team process (and related team structure). This computational SMM plays a critical role in allowing CAST to find a desired balance between achieving efficiency and maintaining adaptability in teamwork modeling. Efficiency of teamwork is realized by (1) anticipating information needs of teammates using the SMM, (2) adopting a decision-theoretic strategy for proactive communications based on the SMM, and (3) rationally fuse information to meet others' higher-level information needs. Adaptability of the team is realized by dynamically assigning tasks among agents (which updates the SMM) based on the SMM regarding their roles and other constraints specified in the team plans.

Due to the rapid advancement of information technologies, the amount of information that needs to be analyzed, interpreted, shared, and fused by a team has been increasing at a speed never seen in human history. This trend has begun to raise the challenges of information sharing, collaboration, decision support for teams to a new level. Empowering software agents with a better understanding about the behavior of team members (including humans) can become increasingly important for developing solutions for addressing these challenges.

Acknowledgments

This research is supported by a DOD MURI grant F49620-00-1-0326 administered through AFOSR.

References

[1] E. Blickensderfer, J. A. Cannon-Bowers, and E. Salas. Cross-training and team performance. In J. A. Cannon-Bowers and E. Salas, editors, *Making Decisions Under Stress*, pages 299–311. Washington DC: American Psychological Association, 1998.

[2] R. R. Brooks and S. S. Iyengar. *Multi-Sensor Fusion: Fundamentals and Applications*. Prentics Hall, New Jersy, 1998.

[3] J. A. Cannon-Bowers, E. Salas, E. L. Blickensderfer, and C. A. Bowers. The impact of cross-training and workload on team functioning: A replication and extension of initial findings. *Human Factors*, 40:92–101, 1998.

[4] J. A. Cannon-Bowers, E. Salas, and S. A. Converse. Cognitive psychology and team training: Training shared mental models and complex systems. *Human Factors Society Bulletin*, 33:1–4, 1990.

[5] J. A. Cannon-Bowers, E. Salas, and S. A. Converse. Shared mental models in expert team decision making. In *Individual and group decision making*, pages 221–246. Castellan, NJ, 1993.

[6] P. R. Cohen and H. J. Levesque. Teamwork. *Nous*, 25(4):487–512, 1991.

[7] M. D. Coovert and K. McNelis. Team decision making and performance: A review and proposed modeling approach employing petri nets. In R. W. Swezey and E. Salas, editors, *Teams: Their Training and Performance*. Ablex Pub Corp., 1992.

[8] B. Grosz and S. Kraus. Collaborative plans for complex group actions. *Artificial Intelligence*, 86:269–358, 1996.

[9] B. Grosz and S. Kraus. The evolution of sharedplans. *Found. and Theories of Rational Agencies*, pages 227–262, 1999.

[10] E. J. Horvitz, J. S. Breese, and M. Henrion. Decision theory in expert systems and artificial intelligence. *Journal of Approximate Reasoning, 2(Special Issue on Uncertainty in Artificial Intelligence)*, pages 247–302, 1988.

[11] T. R. Ioerger. Jare: Java automated reasoning engine.

[12] J. Orasanu. Shared mental models and crew performance. In *Proceedings of the 34 Annual Meeting of the Human Factors Society*, Orlando, FL, 1990.

[13] C. Porter, J. Hollenbeck, D. Ilgen, A. Ellis, B. West, and H. Moon. Towards understanding backing up behaviors in work teams: The role of personality and the legitimacy of need for backing up others. *Journal of Applied Psychology*, page in press, 2003.

[14] G. M. Powell and B. Broome. Fusion-based knowledge for the objective force. In *National symposium on Sensor and Data Fusion*, 2002.

[15] W. Rouse, J. Cannon-Bowers, and E. Salas. The role of mental models in team performance in complex systems. *IEEE Trans. on Sys., man, and Cyber*, 22(6):1296–1308, 1992.

[16] C. E. Volpe, J. A. Cannon-Bowers, E. Salas, and P. Spector. The impact of cross-training on team functioning: An empirical investigation. *Human Factors*, 38:87–100, 1996.

[17] J. Yen, X. Fan, and R. A. Volz. On proactive delivery of needed information to teammates. In *Proc. of the Workshop on Teamwork and Coalition Formation at AAMAS'02*, pages 53–61, July 2002.

[18] J. Yen, X. Fan, and R. A. Volz. Proactive information exchanges based on the awareness of teammates' information needs. In *Proceedings of the AAMAS 2003 Workshop on Agent Communication Languages and Communication Policies*, Melbourne, Australia, 2003.

[19] J. Yen, J. Yin, T. Ioerger, M. Miller, D. Xu, and R. Volz. Cast: Collaborative agents for simulating teamworks. In *Proceedings of IJCAI'2001*, pages 1135–1142, 2001.

[20] J. Yin, M. S. Miller, T. R. Ioerger, J. Yen, and R. A. Volz. A knowledge-based approach for designing intelligent team training systems. In *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 427–434, 2000.