To appear in *Decision Support Systems*: Special issue on Intelligence and Security Informatics

Agents with Shared Mental Models for Enhancing Team Decision-Makings[1]

John Yen[1], Xiaocong Fan[1], Shuang Sun[1], Timothy Hanratty[2], and John Dumer[2]

[1] *School of Information Sciences and Technology*
The Pennsylvania State University
University Park, PA16802

[2] *US Army Research Laboratory*
Aberdeen Proving Ground, MD 21005

Abstract

Proactive information sharing is a challenging issue faced by intelligence agencies in effectively making critical decisions under time pressure in areas related to homeland security. Motivated by psychological studies on human teams, a team-oriented agent architecture -- CAST (Collaborative Agents for Simulating Teamwork), was implemented to allow agents in a team to anticipate the information needs of teammates and help them with their information needs proactively, effectively, and timely. In this paper, we extend CAST with a decision-making module. Through two sets of experiments in a simulated battlefield, we evaluate the effectiveness of the decision theoretic proactive communication strategy in improving team performance, and the effectiveness of information fusion as an approach to alleviating the information overload problem faced by distributed decision makers.

Keywords : Homeland Security, Information Overload, Agent Teamwork, Team Decision-making

1. Introduction

A team of knowledge workers in a homeland security (HS) area face many challenges in their information processing and decision makings for detecting potential terrorist threats, preventing them from occurring, and/or responding quickly and appropriately to terrorist attacks that actually occurred [28]. First, they need to process, interpret, and analyze a huge amount of highly dynamic information. Second, due to the broad scope of homeland security issues, the knowledge and expertise for interpreting and fusing information related to homeland security is often *distributed* among a HS decision making (DM) team. For example, one member of the team may be an expert in terrorist organizations, while another is an expert in biological threats. Third, members of the team may have different access to various information sources due to security concerns or due to their role and responsibility in the team. For example, an analyst may have access to satellite images while another analyst has access to intelligence reports. These three

[1] **Revised paper for DSS special issue**. Send proofs to John Yen, School of Information Sciences and Technology, The Penn State University, PA 16802.

challenges can significantly hamper the quality and the timeliness of decision makings in homeland security areas, which can have shocking consequences.

Several existing technologies can be leveraged to address some of these challenges. For example, multiple heterogeneous information sources can be integrated to enable the processing of queries that pull information from multiple sources. Search engines (especially those specialized in certain areas) and advanced information retrieval techniques can be used to find and filter information from the Web and other text documents. While these techniques can be part of a solution, they do not address the challenge of helping a HS DM team, with different knowledge and information access, to better *collaborate* with each other for effective team decision makings.

Obviously, the identification of cognitive factors that contribute to high-performance human teams is critical for investigating theories, models, algorithms, and software implementations that can simulate these cognitive factors to assist human DM teams. Psychological studies about human teams have repeatedly point out that high performance human teams share certain characteristics, which include the following:

- They can often anticipate needs of other teammates;
- They can proactively help teammates regarding their needs.

One of the team cognition theories that attempt to explain these teamwork behaviors introduces the notion of “shared mental model” [7], which refers to an overlapping understanding among members of the team regarding their objectives, their structure, their process, etc.

The long-term goal of our research, hence, is to develop theories, models, and agent technologies that can assist a human/agent DM team by proactively delivering needed information to teammates, whether it’s a human or an agent, based on each agent’s computational representation of a “shared mental model” (SMM) about the team. Toward this vision, we have developed an agent architecture called CAST (Collaborative Agents for Simulating Teamwork) [44].

This vision responds to the challenges for HS DM teams in two important ways. First, it helps a team member to receive information that he/she needs but can not access. Second, it enables an agent to fuse information using its knowledge to reduce the cognitive load of a teammate (both in processing and interpreting information). To assess the potential benefits of CAST agent technology in supporting HS teams, in this paper we extend CAST with a decision-making module and carry out experiments using the domain of network-centric warfare (NCW) [1], which shares all the challenges of homeland security team described above. In network-centric warfare, an overwhelming amount of information about the battlefield situations needs to be analyzed. Members of a command and control team in the battlefield (i.e., the operational center of an echelon unit), for instance, often receive different information due to the difference of their sensing capabilities and their location. Finally, the knowledge and expertise are distributed among such a team according to their roles. For instance, an intelligence officer is knowledgeable about assessing enemy threats, whereas a logistic officer is knowledgeable about planning and scheduling supplies. We hope the experiments based on network-centric warfare can provide an initial assessment about the potential utility of applying CAST agent technology to developing cognitive aids for supporting HS DM teams.

The rest of the paper is organized as follows. Section 2 gives the background of shared mental model and related agent technologies. Section 3 briefly describes the CAST architecture. We extend CAST with a decision-making module in Section 4, and report the design and results of two experiments in Section 5. The potential applications of CAST in ISI related domains are further discussed in Section 6, and Section 7 concludes the paper.

2. Background

2.1. Research on Shared Mental Models

Mental models are an internal representation of a situation that has a property of "completeness" at a certain level of granularity. A typical example of a mental model is an image (concrete, explicit), which contrasts with the logical/relational description of a scene (implicit, requires inference), though many other representations of mental models are possible (including vivified databases [3] and Bayesian probability estimates[5]).

Shared mental models are a hypothetical cognitive construct that extends the notion of individual mental models to a team context [29]. A shared mental model produces a mutual awareness, with which team members can reason not only about their own situation, but also the status and activities of the other team members in the pursuit of the joint goals. It has been put forward to explain certain coordinated behaviors of human teams [7][8][29][33][31].

The scope of shared mental models is very broad, which may involve shared ontology, common knowledge and beliefs, joint goals/intentions, shared team structure, common recipes, shared plans, etc. The need for agents to *share ontology* has long been recognized as an important issue for agents to be able to understand each other [12]. A shared ontology provides the common vocabulary for agents to communicate directly. Without a shared ontology, agents can communicate only through a "broker" who provides translations between different ontologies. Efforts (e.g., DARPA Agent Markup Language[12] [14]) have been made to facilitate sharing ontology on the semantic web.

Common knowledge (beliefs) of the domain problem under concern, the communication protocols to use, the assumptions to take (e.g., agent sincerity, communication reliability, etc.) establishes a common basis for agents to understand and respond to each other's behaviors. A team structure may specify such information as membership of a team, sub-team relations, pre-determined team leader, roles each member can play, capability requirements on each role and so forth [43]. To have a shared team structure enables an agent to develop a higher level abstraction about the capabilities, expertise, and responsibilities of other team members.

Most collaborating agents have *shared goals*, either explicitly or implicitly. Joint intention theory [10] formally defines the meaning and implications for multiple agents to commit to a shared goal (i.e., joint intention). 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.

Shared knowledge about the *process* of a team provides the 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". The main advantage of having a process shared is that it forces the consequences of a situation to be worked out and realized, and allows predictions to be made, which can be very useful for anticipating what is likely to happen in the future. The work most related to shared team process are the notion of common recipe in the Joint Responsibility model [23] and the notion of shared plans in the SharedPlans framework[15] [16]. Common recipe provides a context for the performance of actions in much the same way as the joint goal guides the objectives of the individuals [23]. A shared plan is characterized in a mental-state view as a particular collection of beliefs and intentions; an agent is said to have a shared plan with others if and only if the agent works towards establishing and maintaining those required mental attitudes, and it believes the other agents do so likewise [17]. The SharedPlans theory provides a basis for formalizing the proactive information delivery behavior of team agents [41].

Shared mental models can be measured in terms of the degree of overlap or consistency among team members' knowledge and beliefs [8]. Due to the importance of shared mental models to teamwork, fostering the development of shared mental models has been the target of several practical methods for training teams. 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 and to detect inconsistencies or changes early on. A popular method aimed at increasing the overlap of static knowledge (e.g. of roles and responsibilities) is cross-training [2][6][39].

2.2. Research on Agent Teamwork

Teamwork has been the focus of a great deal of research in Distributed Artificial Intelligence field. Joint Intention theory and SharedPlans theory are two widely-accepted formalisms for teamwork. To distinguish team behavior from coordinated individual behavior (i.e., individuals' goal happens to be the same), a notion of joint mental attitude—joint intention, is introduced based on the concept of joint persistent goal [10]. Joint intentions can be viewed as a joint commitment to perform a collective action to achieve a certain joint goal. Joint intention theory is significant because it not only offers a framework for studying numerous teamwork issues, but also provides a guide for implementing multi-agent systems.

The SharedPlans formalism originates from Pollack's mental state model of plans [30]. In [15][16], Grosz and Kraus formalized the concepts of individual plans and shared plans, and explored how a shared plan for a team action evolves from partial (possibly with partial recipe) to a final complete form with complete recipes for the team action and all the subsidiary actions at different levels of abstraction. In evolving a shared plan, although the common recipe tree becomes complete from external, team members may only possess different partial views of the tree. In pursuing their common goals, it is the shared plans that ensure team members to cooperate smoothly rather than prohibiting each other's behavior, which may occur otherwise due to the partial views of the common recipe tree.

Several computational models of teamwork have been developed for producing cooperative behaviors among intelligent agents. GRATE* is an implemented system based on the Joint Responsibility model[23]. Mainly focusing on handling unexpected failures, the Joint Responsibility model refines Cohen and Levesque's joint intention theory by explicitly capturing different causes of recipe failures. Specifically, it clearly specifies the conditions under which an agent involved in a team activity should reconsider its commitments (three related to joint goal commitment: goal has been attained, goal will never be attained or goal motivation no longer present; four related to common recipe commitment: desired outcome is available, or recipe becomes invalid, untenable or violated). The model furthermore describes how the agent should behave both locally and with respect to its fellow team members if any such situations arise: it should drop the commitment and must endeavor to inform all the other team members so that futile activities can be stopped at the earliest possible opportunity. In GRATE*, an organizer (leader) agent plays a critical role in carrying out a team activity. It is responsible for detecting the need for joint actions, for establishing a team and ensuring members' commitments are as required by the team's joint responsibility.

STEAM [37] is a hybrid teamwork model that borrows from the strengths of both the joint intention theory and the SharedPlans formalism. It uses joint intentions as a building block to hierarchically build up the mental attitude of individual team members, and ensure that team members pursue a common solution path. STEAM exhibits two valuable features: selective communication and the way of dealing with teamwork failures. In STEAM, communication is driven by commitments embodied in the joint intentions theory, as well as explicit declaration of information-dependency relationships among actions. To make a decision on communication, STEAM agents take into consideration the communication costs, benefits, and the likelihood that some relevant information may be already mutually believed. To handle failures, STEAM uses *role-monitoring constraints* (AND, OR, dependency) to specify the relationship of a team operator and individual's or subteam's contributions to it. When an agent is unable to complete actions in its role and the embedding team operator is still achievable, the remaining agents will invoke a repair plan accordingly.

COLLAGEN [34] is the first implemented system based on the SharedPlans theory. It incorporates certain algorithms for discourse generation and interpretation, and is able to maintain a segmented interaction history, which facilitates the discourse between human user and the intelligent agent. Another team-oriented agent architecture, which is based on the RETSINA individual agent architecture [36], is proposed in [13]. A partial plan (which is a high-level description of the mission under concern and resembles a SharedPlans recipe) is initially shared among the team-oriented agents. This plan is continually updated and revised throughout the execution of the mission; such a process is reminiscent of the SharedPlans' partial plan refinement process. In addition, they explicitly recognized the notion of checkpoints as a mechanism by which teammates can communicate and verify the progress of commitments to a goal.

3. CAST--Collaborative Agents for Simulating Teamwork

CAST is a teamwork model that enables agents, whether they are software agents or human agents, to anticipate potential information needs among teammates, and exchange information proactively [44]. It has been designed to study teamwork-related issues emerging from teams with well-defined structure and process, distributed expertise, and limited communication in time-stress domains. Fig.1(a) depicts a team of CAST agents, where some of them interact with human users as assistants. Briefly, the key features that distinguish CAST from the aforementioned team-oriented agent architectures include:

- CAST employs a richer generic language for encoding teamwork (recipe) process, where the constraints for task assignments, preconditions of actions, decision points within a process and termination conditions of a process can be specified explicitly. The other agent systems either lack an explicit encoding language or have a language that is limited in describing complex team processes;
- CAST is empowered with a richer notion of shared mental model—shared team process (including both static team recipe and dynamic team process);
- CAST agents, based on the richer SMM, are able to dynamically anticipate teammates' relevant information-needs and proactively deliver the needed information to them.

3.1. The CAST Architecture

As shown in Fig.1(b), an CAST agent is composed of six components: *Reasoning Engine* (RE), *Shared Mental Model* (SMM), *Individual Mental Model* (IMM), *Team Process Tracking* (TPT), *Proactive Behavior* (PB), *and Goal Management* (GM) (we will extend CAST with a *decision making* module in Section 4). Briefly, the RE, based on the current states of SMM and IMM, triggers appropriate algorithms in TPT, PB and GM to monitor the progress of team activities, to select goals to pursue, to anticipate others' information needs and proactively help them. The execution of these mental operations will further affect the evolution of the shared and individual mental states.

The TPT module encapsulates two algorithms: *Process-Interpreter &Coordinator* (PIC) and *Dynamic-Agent-Assignment* (DAA). In CAST, team processes are originally coded in a high-level language called MALLET (Multi-Agent Logic-based Language for Encoding Teamwork), a logic-based language for specifying the structures and processes of agent teams [43]. To facilitate dynamic reasoning and monitoring, team processes are internally represented as Predicate-Transition (PrT) nets [11][40], which are generated offline by a MALLET Parser. The *PIC* algorithm is used by individual agents to interpret and manipulate a team process so that they could collaborate smoothly both when everything is progressing as planned and when something goes wrong unexpectedly. More specifically, in normal circumstances, *PIC* ensures all the team members behave, react, and deliberate strictly according to the committed (intended) plans (i.e., courses of actions), and synchronize their behaviors whenever necessary. When agents encounter in exceptional circumstances, they use their *PIC*s to collaboratively adapt to changes in the environment.

Initially, the internal PrT nets are only partially instantiated (i.e., with the actors of some tasks to be resolved at run time). Each unresolved task can be associated with certain constraint conditions (e.g., requirements on roles, agent workload, etc.). The *DAA* algorithm is responsible for initiating communications at appropriate time to determine suitable actors for those unresolved tasks in the team process. The result of the *DAA* will further affect an agent's reasoning about others' information needs.

The PB module encapsulates all the proactive teamwork capabilities. One algorithm in PB, called DIARG (Dynamic Inter-Agent Rule Generator), implements the proactive information delivery behavior as exhibited in effective human teams. The foundation of *DIARG* algorithm has been established by extending the SharedPlans theory with the formal semantics of proactive communicative actions [41][42]. Dynamic-Information-Flow-Table (DIFT) is a data structure related to *DIARG*. Each agent generates an internal *DIFT* at compile time by extracting the potential information needs from the team process specified in MALLET. *DIFT* can be dynamically updated to reflect the most current information-needs relationships among the team, and is used by *DIARG* to anticipate teammates' information-needs. Upon acquiring new information from the environment, *DIARG* checks *DIFT* to see whether the new information matches some teammates' future information-needs. If there is a match, the agent will consider sending out the new information to the corresponding needers proactively.

An agent may have multiple goals. Some are individual goals and some are team goals; some may have temporal relations while some may not. The *GM* module is used by a CAST agent to select a goal to pursue, or suspend the pursuit of one goal and switch to another; both are based on the agent's situation assessment and cooperation requests from other agents. Once a goal is committed, *GM* will find a plan that can achieve the goal; the PrT net generated for the plan will become the agent's work process.

IMM stores those mental attitudes privately held by individual agents. In its IMM an agent may maintain its inference knowledge (e.g., Horn-clauses), its own beliefs about the dynamic world, beliefs about the capability of others, or even beliefs about the beliefs of others. It is continuously updated by sensor inputs and communication messages received from other agents. CAST agents currently employ an embedded JARE (Java Automated Reasoning Engine)[22] as the inference engine and the manager of IMMs. An agent will cast all the queries (for checking conditions or constraints) to, and wait for answers from its JARE. CAST supports belief reasoning of other teammates by inferring from their observability, or from the observable effects of actions already completed. This is important for CAST agents to be able to anticipate other teammates' information needs, which highly impacts the information exchanges among team members. For instance, suppose crime analyst *A* can observe the behavior of a suspect if *A* and the suspect is in the same building. Given information about the location of *A* and the suspect, a teammate of *A* (who is equipped with advanced locating systems) can infer whether *A* can observe the suspect. Then, the teammate may choose to inform the location information of the suspect to *A* proactively if he infers that *A* cannot observe it.

3.2. The Computational Shared Mental Model in CAST

The SMM stores the knowledge and information that are shared by all team members. It has four components: team process, team structure, domain knowledge, and *DIFT*. 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 more like incomplete recipes in the SharedPlans theory, since the performers of unresolved tasks need to be determined at run time. 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 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 *DIFT* component maintains the dynamic information-needs relationships (i.e., make sure the information needs reflect the current status of team activities).

Currently CAST supports three kinds of information-needs. First, CAST is implemented such that each team member commits 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 the dynamic status of team process. It is such *built-in* information-needs that provide the cohesive force [9]that binds individual CAST agents together as a team. On the one hand, as information-needs they are part of the shared mental model; on the other hand, they are leveraged to maintain the other part (progress of team process) of the shared mental model. The second kind is those information-needs explicitly *coded* in a team process. CAST agents can extract the pre-conditions, termination conditions and constraints associated with (sub-)plans in a team process, and establish partial information-flow relationships based on incomplete knowledge. These partial relationships can be further refined at run time as the team do dynamic agent assignments. The third kind of information-needs is those explicitly *communicated* among teammates.

In our opinion, SMM is different from mutual beliefs (MB) in that 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 the differences that may emerge. For instance, when an agent finishes a task step, the agent will inform others that it will proceed to the next. 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[10].

4. Extend CAST to support decision-makings

The purpose of maintaining the SMM among team agents is to allow agents to leverage the computational shared mental model to achieve effective teamwork. As already mentioned, the SMM is used in algorithm *DIARG* to anticipate others' information needs. In this section, we extend CAST with a *Decision-Making* module to allow CAST agents to make decisions on (1) the next course of actions when facing a choice point; (2) whether to inform others when obtaining a new piece of information; and (3) whether to fuse information before delivery when the recipient's cognitive capacity is considered. SMM is critical in making all of these kinds of decisions.

4.1. Making decisions at explicit decision points

The MALLET language has a CHOICE construct, which can be used to specify explicit choice points in a complex team process. For example, suppose a fire-fighting team is assigned to extinguish a fire caused by an explosion at a chemical plant. After collecting enough information (e.g., chemicals in the plant, nearby dangerous facilities, etc.), the team needs to decide how to put out the fire. They have to select one plan if there exist several options.

The choice construct is composed of a list of branches, each of which specifies a plan (a course of actions) and may be associated with preference conditions and a priority information. The preference conditions of a branch is a collection of first-order formulas; the evaluation of their conjunction determines whether the branch is workable under that context. The priority information is used in selecting a branch in case that the preference conditions of more than one branch are satisfiable.

The following algorithm is implemented in CAST to allow an agent to make decisions on the next course of actions the team should take.

```
Algorithm 1: CPDDM () /*choice point decisions by designated decision maker*/
    1. Upon reaching a choice point, each agent checks if itself is the designated decision
       maker;
    2. If it is, the agent evaluates the preference conditions of the potential alternatives based on
       the information available currently. The branches whose preference preconditions is
       satisfiable are marked as workable.
           a. If there is only one workable branch, this branch is committed;
           b. If there are more than one workable branch, the branch with the highest priority
              is committed. If there are more than one branch having the highest priority,
              randomly select one;
           c. If there is no workable branch, wait one step (more information becomes
              available) and restart CPDDM();
           d. Inform the other teammates of the chosen branch, and skip to step 4;
    3. If not, wait until being informed of the choice from the decision maker; In the meantime,
       anticipate the information-needs of the decision maker and proactively deliver
       information that can improve its situation awareness for making better decisions;
    4. All team members perform the selected course of actions; In the meantime, in case some
       agent detects the execution should be terminated, it will let others know so that all the
       agents involved in the current team activity can backtrack to the last choice point, and
       choose another COA if needed.
```

This collaborative decision-making algorithm requires a team member be designated as the decision maker, who on behalf of the whole team makes the final decisions. Such leader based team decision making is well adopted in practical systems (e.g., [16]) due to its simplicity: no negotiation among teammates is needed to compromise their individual decisions. Furthermore, The algorithm prescribes a decision-making process that emphasizes on the proactive information sharing rather than the computational logics. Hence, it can be easily extended to support naturalistic decision makings (e.g., Recognition-primed decision making model[25]).

4.2. Making decisions on proactive communication

An agent will consider proactively sharing a piece of information with other agents if it believes this will allow them to make better decisions or prevent them from getting stuck in action performing. However, when communication cost is not neglectable, proactive information delivery may not always benefit the whole team. For instance, the messages to teammates may be overheard by agents in an opposite team, who may change their tactical strategies accordingly to attack the first team. To avoid this, an agent should evaluate the tradeoff between the benefits and the potential costs of sharing information before actually doing it.

Generally speaking, factors affecting a decision on proactive communication can be grouped into two categories: (1) communication costs to self and (2) impacts 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 sensing 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.

We extend CAST with a decision-theoretic communication strategy which considers communication costs and the uncertainty of the different possible outcomes in calculating the expected utility of "sending a piece of information". A similar calculation can be made for the expected utility of not sending the information. Agents simply choose the alternative with the higher expected utility [20]. In other words, if the expected utility of sending information exceeds the expected utility of not-sending, a CAST agent will choose to proactively deliver the information to the needers, believing the potential benefit outgoes the communication cost.

We use *need(B,I)* to denote the fact that agent *B* will need information *I* (in some future), and assume that agent *A* believes *need(B,I)* in case that (1) *A* believes *B* potentially intends to do some action (in the near future) and *I* matches with some conjunct of the action's pre-condition; or (2) *A* believes *B* potentially intends to do some action (in the near future) and *I* matches with some conjunct of the action's termination-condition; or (3) *A* believes *I* is essential for *B* to pursue a goal, i.e., not having information *I* (e.g., threat information) endangers the achievement of the goal *B* is pursuing. Thus, from *A*'s viewpoint, the probability of *need(B,I)*, denoted as p, depends on the probability of *A*'s beliefs about *B*: the probability τ of B's holding of the intention to do an action where *I* is relevant to the preconditions or termination conditions of the action, the probability σ of *B* having a goal (intention that a proposition be true) g, and the probability δ of not sending *I* to *B* endangers the achievement of goal g. We also assume the probability of *need(B,I)* is independent of the probability of "*B* believes *I* (now)".

Figure 2 shows the decision tree for an agent *A* to decide whether to proactively send information *I* to a teammate *B*, where rewards and costs are measured with respect to the whole team. There are two choices: *PI* (inform with communication cost *Cc*) and *NPI* (not inform, communication cost is 0). There are three possible outcomes if agent *A* chooses *PI*. The first possible outcome is that *need(B,I)* holds with probability *p* and *B* does not believe *I* with probability *q*. This is the ideal case and the reward to the team is *U(I)*. The second outcome is that *need(B,I)* holds with probability *p* and *B* already believes *I* with probability *(1-q)*. In such a case the reward to the team is *U(I)-Cm*, where *Cm* is a penalty for sending information that *B* already knows. The third outcome is that *B* does not need *I* with probability *(1-p)*; the reward to the team is *U(I)-Cn*, accounting for the penalty of futile communication (mis-coordination). There are also three possible outcomes if agent *A* chooses *NPI*. The first possible outcome is that *need(B,I)* holds with probability *p* and *B* does not believe *I* with probability q. This is the worst case and the team takes the most penalty for mis-coordination, so the reward is *U(I)-Cn*. The second outcome is that *need(B,I)* holds with probability *p* and *B* already believes *I* with probability *(1-q)*; this is an ideal case and the reward to the team is *U(I)*. The third outcome is that *B* will not need *I* with probability *(1-p)*; the reward to the team is *U(I)* for frugality in communication. The final choice of *PI* or *NPI* is based on their expected utility. The expected utility *EU(PI)* of *PI* is $U(I)-Cm \times p \times (1-q)-Cn \times (1-p)-Cc$, while the expected utility *EU(NPI)* of *NPI* is $U(I)-Cn \times p \times q$. The agent will inform *I* to *B* iff $EU(PI)>EU(NPI)$, i.e., iff $(Cn+Cm)p \times q-Cm \times p>Cc+(1-p)Cn$. Compared with the decision tree given in [37], the tree shown in Figure 2 is more general: the consideration of the uncertainty of an event's threat to a joint intention is just one special case in reasoning about the probability of *need(B,I)*.

The costs (*Cn*, *Cm* and *Cc*) are assumed to be domain knowledge, and can be encoded as part of the Shared Mental Model maintained by all the agents in a team. To compute the expected utilities, we still need to estimate the probabilities *p* and *q*. As discussed above, the probabilities τ, σ and δ are needed in estimating the probability *p* of *need(B,I)*. As a first step, CAST agents estimate *p* based on the following criteria: (a) the probability (τ) that B intends to do a certain action is high if according to the SMM agent *B* is assigned to do the action or is one of a few candidates; τ is low if according to the SMM agent B's workload is high and the number of candidate doers of the action is large (e.g., more than 3); (b) the probability σ of *B* having a goal is high if the performance of its current activities will bring B closer to the goal; it is low otherwise; (c) the probability (δ) that *A* not sending *I* to *B* will endanger the achievement of a goal is high if the goal becomes unattainable if *B* lacks *I*. Obviously, δ is inversely proportional to the number of team members who can sense *I*; (d) the probability *p* is proportional to τ, σ and δ, which one is used in estimating *p* depends on the actual situations.

The probability *q* that *B* does not believe *I* is evaluated based on *A* and B's observability and the dynamics of information *I*. If *A* has observed that *B* is doing another action that also requires *I*, *A* will assume that *B* already knows *I* with a higher probability. If as a prior knowledge *A* knows *B*'s observability, and from the current context *A* believes that *B* could observe *I*, then *A* will assume *B* knows *I* with a higher probability. If *I* is a static information, and *A* remembers that itself once informed *I* to *B* or itself actually got to know *I* from *B* before, the probability that *B* believes *I* is 1. In other cases, *A* will assume the probability that *B* believes *I* is very low.

4.3. Make decisions on fusing information

The long term goal of CAST is to facilitate the investigation of teamwork problems involving human teammates (e.g., a team of homeland security analysts, assisted by CAST agents, collaboratively make decisions on the potential threats). It is well understood that human has limited cognitive capacity [16]. To enable an CAST agent to support human team members, it has to take into its consideration the human recipient's cognitive load, and proactively fuse information, if necessary, before delivering any information to the human. The decision on whether to send multiple lower-level information or fused higher-level information to a teammate could be fairly simple: just compare the amount of information to be sent and the reserved capacity of the recipient, fuse if the former is larger and not fuse otherwise. The decision could be complicated if other factors (e.g., the long term impact of information fusion on the recipient's cognitive capacity) are considered.

In doing this, we first need an appropriate way to represent the inference-level contexts of information-needs such that it can facilitate an agent not only to assist teammates even when it only has partial information relevant to their needs, also to alleviate teammates' information overload by delivering fused information that is in a form closest to their information-needs. Towards these objectives, we transform agents' inference knowledge (represented as Horn-clauses in CAST) to hierarchical inference trees.

An inference tree includes two kinds of nodes: "AND" nodes and "OR" nodes. Each "AND" node has several "OR" nodes as its sons; while each "OR" node is labeled with a predicate, and may have several "AND" nodes as its sons. The evaluation of an "AND" node is true iff the conjunction of those predicates labeling its son nodes is true; the evaluation of an "OR" node is true iff any of its son nodes is true. Every "AND" node, if it has a parent node, represents one piece of inference knowledge (i.e., horn-clause): the truth value of the predicate labeling its parent can be inferred from the truth values of the predicates labeling its son nodes.

Take digital battlefield as an example. Suppose the preconditions of plan *removeThreat* has three conjuncts: *threat(?e,?loc,?dir,?num)*, *dir(?e,?dir)*, and *canFight(Self)*, which are the head predicate of the horn-clauses in Table 1, respectively. That is, an agent could deduce the existence of a threat if it has beliefs about the identified enemy unit (*IsEnemy*), the location of the enemy unit (*At*), the moving direction of the enemy unit (*Dir*), and the number of enemies in the unit (*Number*); to deduce the moving direction, the agent needs to know the change of location; and to be able to fight, the agent needs to have enough fighting power, and it can move to the targets. The inference tree of plan *removeThreat* is shown in Fig.3.

In an inference-tree, the nodes at the same level collectively form a context for each individual. For instance, in Fig.3, as far as threat identification is concerned, *Dir(?e,?dir)* is useful only when it is evaluated together with *IsEnemy(?e)*, *At(?e,?loc,NOW)*, and *Number(?e,?num)*. Thus, these four predicates collectively establish a context for each of the individual predicates.

Given a collection of predicates and related inference knowledge, its inference tree can be generated recursively. Here we simply assume the information-needs (pre-conditions and termination conditions) associated with each plan involved in the team process are all organized as inference trees, and see how inference trees can be used in agent teamwork settings.

First, inference trees can be used in *collaborative constraints satisfaction*. Suppose agents *A1*, *A2* and *A3* share the inference tree as shown in Fig.3, and *A3* is the doer of plan *removeThreat*. Assume both *A1* and *A2* have identified an enemy unit (*e1*) approaching *A3*, who is unaware of the threat from *e1*. Also assume *A1* can only observe the location, *At(e1,area_4,NOW)*, and moving direction, *Dir(e1,northeast)*, of *e1*; *A2* can only observe the enemy number, *Number(e1,100)*, of unit *e1*. Obviously, neither *A1* nor *A2* alone can enable *A3* to do *removeThreat*. However, they can collaboratively satisfy *A3*, because *A1* knows *At(e1,area_4,NOW)* and *Dir(e1,northeast)* will be useful for *A3* in the context indicated by the dotted circle in Fig.3, and *A2* knows *Number(e1,100)* will be useful for *A3* in the same context.

Another important use of inference trees is that they can be leveraged to maximally assist teammates' information-needs without overloading them. The following algorithm shows how inference trees can be used to fuse lower-level information to satisfy others' higher-level information-needs.

```
Algorithm 2: info_fusion(ITNode root)
  doerList = getAssignedDoers(root.plan())
  fusionList = null; /*a list of OR nodes*/
  if (root is AND node)
    fusionList.addAll(root.son());
  else fusionList.add(root);
  while (fusionList is not null) and (doerList is not null)
    onode = fusionList.removeFirst();
    result = agent.check(onode.predicate()); (1)
    if (result is true)
      for each doer in doerList                (2)
        if ((preferFusion(doer) is true) or
          (onode is a leave node))
          agent.proTell(doer, onode.predicate());
          doerList.remove(doer);
      end {for}
    end {if}
    //process lower-level information
    for each son anode of onode                (3)
      fusionList.addAll(anode.son());
  end {while}
end.
```

Every time after an agent senses the environment, it will first check the sub-plans to be done and collect the inference trees corresponding to those sub-plans whose pre-conditions or termination conditions are related with the new sensed information. The algorithm *info_fusion* will be invoked for each of the collected inference trees. *info_fusion* first gets all the agents assigned to do the sub-plan, then employs a breath-first reasoning. For each OR node in *fusionList* the agent checks whether the associated predicate holds or not through its reasoning engine (clause 1). If it holds, for each *doer* remained in *doerList* this agent decides, based on *doer*'s cognitive load, whether to inform *doer* a piece of fused information or multiple pieces of lower-level information; if fused information is preferred for *doer* or the predicate cannot be further decomposed, this agent simply delivers the fused information to *doer* and removes *doer* from the *doerList* (so that *doer* will no longer be considered in the successive iterations). Since some agents remained in *doerList* may prefer lower-level information, in clause (3) the next level OR nodes are added to the *fusionList* for further processing. Suppose the team size (the number of agents in the team) is n, the number of predicates referred to in Horn-clauses is m (this means an inference tree at most has m OR nodes). Then, in the worst case, $|doerList|=n$ and $|fusionList|=m$, the worst case complexity of *info_fusion* is $O(n \times m)$.

The algorithm *info_fusion* enables CAST agents to support the reasoning of information-needs at multiple levels. Note that *info_fusion* is different from the DIARG algorithm in [43] . DIARG considers each newly sensed information separately, while *info_fusion* considers all the new information together as long as they are related (i.e., by certain horn-clauses). Information-fusion can also be carried out by depth-first reasoning. One difference is that breath-first algorithm guarantees that the higher-level information will always be delivered with higher priorities than the lower-level information. This is critical in case that the information consumers are human, who only have limited cognitive capacity [16].

5. Experiments

To evaluate the impacts of the SMM-supported decision-makings on agent 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 the correctness of tactical decision makings when the decision-maker's cognitive capacity is under consideration. Both experiments are carried out in a simulated battlefield, where two opposing agent teams, a friendly team (blue) and an enemy team (red), navigating in the combat field to achieve certain team goals. The agent teams are designed based on four criteria: (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, and (4) the team has a well-defined structure and process. For simplicity, we ignore the need for agents to maintain the security of combat accomplishments (e.g., ground gained).

5.1. Experiment I: The Decision Theoretic Communication Strategy

5.1.1. 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, while the red team tries to protect their base by attacking any approaching blue agents. The red team involves three types of agents: one agent protecting a waypoint on the blue team's attack trajectory, one agent protecting the homebase, and other agents patrol around the homebase. Each agent in the red team has a sensing range. When a red agent detects a blue agent, it will move toward and attack the blue agent. To ensure that the communication cost is not negligible, we designed the scenario to introduce a risk factor into an agent's communication: a blue agent's location can be detected by the enemy base if the agent communicates with its teammates within the "detection ring" of the enemy base. Once perceiving the location of a blue agent, the enemy base will inform this information to a red team agent who is the closest to the intruder (for initial assignment) or to a red team agent who has been previously assigned to the intruder (for tracking and attacking). The red agents act independently and there is no communication among them, because they do not have a shared mental model.

The blue team is formed by agents with three different roles: the scout, who can sense (the sensing range is further than that of red agents) 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 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. More specifically, being informed of 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 of the location of an approaching red agent, a dynamically assigned fighter (based on the team's SMM about the constraints of the assignment and workload) will move toward and attack the red agent. Meanwhile, if a bomber gets such information it will try to move away from the red agent to avoid threat.

In this experiment, we designed two communication strategies for the blue team to evaluate the effectiveness of the decision-theoretic approach for communication. The two strategies differ in how they handle the communication decisions on whether to proactively inform bombers and fighters about red agents' location. Scout agent using strategy $S1$ always informs the closest bomber about the detected red agents so that the bomber can escape; and always informs fighters about red agents' location so that the fighters can move toward and attack the enemies. Scout agent using strategy $S2$ (which adopts a decision-theoretic approach presented in Section 4.2) will choose whether to inform teammates about the detected red agents based on expected utilities: it will help teammates only when the expected utility for communicating exceeds the expected utility of not communicating.

We use the number of enemy units as the control variable, which indicates the level of domain complexity. The more enemy units protecting the waypoint and the homebase, the more difficult for the blue team to succeed. The number of enemy units ranges from 1 to 6. In case that a configuration has two enemy units, they patrol around the enemy base and the waypoint respectively. In case that a configuration only has one enemy unit, the enemy simply patrols around the enemy base. For each fixed number of enemy units, we generated 40 configurations by randomly assigning initial positions for the blue and red agents.

5.1.2. Result Analysis

Fig. 4(a) summarizes the number of successfully completed missions (out of a total of 40 missions) for the blue team using the two communication strategies, which shows strategy *S*2 outperformed strategy *S*1. The performance difference between the strategies was more significant when the number of enemy units increased. These experimental results suggests that the decision-theoretic communication strategies can be effective for team-based agents to decide on whether to proactively deliver needed information to teammates in the circumstances that communications carry a cost. To gain a better insight about the results, Fig. 4(b-d) 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 protecting 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. 4(c). In contrast, the survival rate of scout agent decreased rapidly under the alternative communication strategy (S1). These observations suggested that considering the risk factor of agent communications help protect the scout agent, which contributes to higher chance of mission success.

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

The purpose of this experiment is to understand how significant information fusion is to team performance, especially to the accuracy of decision-making tasks, when the information consumer only has limited cognitive (information processing) capacity. The long-term goal of this study is to explore solutions to the dilemma emerging in homeland security and BattleSpace InfoSphere: decision makers can make better decisions if more relevant information is available, while they can be easily overloaded with overwhelming amount of information.

5.2.1. Scenario and Experiment Design

In this experiment, the blue team adopts a defensive strategy. The scenario is as shown in Fig. 5, where 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 and priorities 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.
- S4 has as its objective the identification of supply routes and sustainment of supplies.

The scenario is marked by 5 events.

- Event 1: the S2 being alerted to message traffic indicating hostile events occurring around H2 airport where it is believed a neutral indigenous force is located. Noteworthy is the fact that H2 airport is not only a critical supply air avenue, but is within 10 kilometers of the major supply route (MSR).
- Event 2: the S4 informing the S2 that some of its forces have come under fire near the H2 airport.
- Event 3: the S2 (prompted by the objective to monitor enemy location and intent) assigning an unmanned aerial reconnaissance vehicles (UAV) to scout the H2 area for clarification.
- Event 4: the UAV confirming through automatic target recognition software the existence of enemy tanks in the vicinity of H2; confirming the belief that the once neutral force is now with the opposition.
- Event 5: utilizing teaming agents and the underlying shared mental model of each of the BFAs objectives, software agents alert each of the BFAs to the current operational picture and advise on possible courses of actions. Noteworthy are the alerts to the S3. Knowing the objectives and priority associated with the S4, the S3-agents are able to anticipate the importance of the situation and take appropriate action.

In this set of experiments we suppose each of S2, S3 and S4 in blue teams are assisted by a CAST agent. The members of the red team are randomly launched and moving nearby the main supply route to introduce potential threats to blue teams. Two blue teams are designed: a fusion team and a non-fusion team. In the fusion team, S2's agent uses inference trees to fuse the information collected by UAVs and other BFAs into higher-level information, and send the fused information to S3's agent (who helps S3 in making decisions). In the non-fusion team, S2's agent simply forwards the relevant information collected by UAVs to S3's agent. Thus, S3 himself has to take some effort in processing the lower-level information before making decisions. At each time step S3 needs to make decisions on whether to fire at the approaching enemies based on the available information and whether the enemies introduce a high-threat to the main supply route. To simulate the behavior of logistics, S4 (through his agent) controls an asset which carries supplies and moves along the main supply route. S4's agent changes the supply route whenever a threat comes. The blue team wins as long as the asset controlled by S4's agent arrives at the pre-specified target.

The following gives a slice of the teamwork knowledge (described in MALLET) shared by the agents in blue teams.

```
(team BlueTeam (S2 S3 S4))
(agent S2)
(agent S3)
(agent S4)
(plays-role S2   (intelligence))
(plays-role S3   (operations))
(plays-role S4   (logistics))
…
(plan handle_route_threat (?t)
(pre-condition (major_supply_route ?r)
       (threat_on_route ?t ?r))
(process
(if (cond
   (cost_of_adjusting_route ?r  low)
   (cost_of_removing_threat ?t high) )
     (DO (S3 S4) (adjust_logistic_route ?r)))
(if (cond
   (cost_of_adjusting_route ?r high)
    (cost_of_removing_threat ?t low) )
      (DO S3 (remove_threat ?t)) )
…
))
```

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. In each run, the number of enemy units and their routes are randomly configured. Each point in Fig. 6(a-b) corresponds to the accumulated results out of 100 runs; 20 runs for each fixed number of enemy units (ranging from 3 to 7). Fig. 6(a) shows the number of successful missions, and Fig. b(b) shows the decision accuracy computed by *(the number of correct decisions)/(total decisions)*. S3's decision is correct if it is the same as the decision 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. 6(c) and Fig. 6(d) corresponds to the accumulated results out of 80 runs, where Fig. 6(c) shows the number of successful missions, and Fig. b(d) shows the decision accuracy.

5.2.2. Result Analysis

According to Fig. 6(a), 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.6(b). Further, in Fig. 6(b) the decision accuracy of the non-fusion team increased as S3' 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. 6(c) and Fig.6(d). 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.

6. Discussion

First, the design of CAST aims at teams with four characteristics: (1) team members have limited and different observability; (2) the team needs to act (e.g., make decisions) under critical time constraints; (3) communication costs are non-negligible; and (4) the team has a well-defined structure and process. These four characteristics are common for many teamwork problems in the real world. For instance, in a naval command and control scenario there are four human officers commanding Carrier, AWACS, Cruiser, and Coastal Air Defense, respectively. They mutually know each other's role, distinct expertise, and the process they have to follow to make team decisions on how to respond to incoming threats [16]. Similarly, a team of anti-terrorism analysts needs to filter, analyze, and fuse overwhelming amount of information from a wide variety of information sources (e.g., satellite images, intelligence reports, radio transmissions, etc.). Each member of these teams can only access a portion of the information accessible to the team, need to make decisions under time pressures, and need to communicate with teammates under resource constraints and/or risks. Teamwork process exhibits itself as collaborative workflows among law enforcement agencies [46], and operational orders (plans) in military.

However, well-defined structure and process does not imply that the structure and process are static. They can adapt to dynamic changes in the environment, as long as such changes are mutually known afterward. Suppose an intelligence cell (S2 officer), an operations cell (S3 officer), and a logistics cell (S4 officer) are involved in a real-world combat and are following a team process where they iteratively carry out a sequence of operations: *gather-info(); measure-threat(); handle-threat()*. Their role in this process is: S2 gathers threat information, S3 needs to measure the level of a threat and all need to react to the threat in *handle-threat()*. Suppose currently the sub-process *handle-threat()* specifies how S2, S3 and S4 should collaboratively remove the incoming threat from the supply route. As time passes, it turns out that the threat cannot be easily removed, then *handle-threat()* needs to be re-adjusted to deal with such a situation (e.g., change the main supply route and make this mutually known). The team needs to be re-structured when a new S2 officer joins the commanding team.

The vision of our research is to empower a team of agents with a computational shared mental model so that they can anticipate needs of teammates and assist them proactively. This vision is not only inspired by psychological studies about human teamwork, but also inspired by growing needs to support mixed human/agent teams in handling overwhelming amount of information under time pressure. Under this vision, CAST is designed with the support of mixed human/agent teams in mind. Due to complexity, we choose to fulfill this vision in two steps. In the first step, team members are pure CAST agents. The CAST agents can serve as assistants to people; in such case people are treated as virtual team members and are invisible to other CAST agents. In this setting, people can share his/her mental models (e.g., goals, interests, human-like decision-making process, etc.) with the assisting agent through an interface. In addition, people can affect the on-going team activities by adjusting the assisting agent's behavior or strategies. As far as decision-making is concerned, having a better sense of the global picture, people can provide their expertise to support the team of agents to make better decisions. On the other hand, an assisting agent can help its user in making better decisions by collaborating with other agents to gather relevant information from distributed intelligence sources. For instance, COPLINK agent provides a Web-based user interface to facilitate the sharing of information in law enforcement among different users [45]. The CAST approach can be applied to the similar scenario, due to the proactive information delivery capability offered by CAST agents. Among others, one of the differences between CAST and COPLINK is that in the CAST approach the collaboration logic is distributed among multiple team agents, which may have access to different intelligence or security sources. The experiments reported in this paper demonstrated our attempt in this first step.

In the second step, a team may have human team members, as well as CAST agents. In this setting, we can still assume each human team member has an assisting agent, but it's the assisting agent rather than the human that is invisible to the other team members. An assisting agent can help in filtering and fusing information that its user receives from other members, in tracking the individual mental model of its user and the mental models being shared among the team, and in forwarding the information to appropriate recipients that its user intends to send. One of our on-going projects is to allow CAST agents and human members to form a decision-making team to respond to incoming threats in the DDD domain [31]. One goal of this project is to investigate how CAST agents can help train human to enhance their teamwork skills.

While the scenarios used in this paper are typical in the NCW domain [1], we would like to use a scenario described in [46]to illustrate explicitly how a team of CAST agents could support a team of people to accomplish a time critical task related to homeland security. Suppose a report of an explosion at a chemical plant is received by a homeland security (HLS) analyst, local firefighters and state police. The assistant agent (AA) for the analyst "reads" the report and immediately retrieves information about what chemicals exist in the plant and a map of nearby dangerous and vulnerable facilities. The firefighters arrive on the scene and identify the sector of the plant where flames and smoke are emanating from. A state police is informed of a suspicious individual in the parking lot. He enters the license plate of a suspicious car in the parking lot and the analyst AA immediately searches for aliases of the owner and links to terrorist organizations. A link is found and the HLS analyst's AA searches for an AA of an expert on that terrorist group. The terrorist expert AA notifies the HLS AA that an associate of the suspicious person is a chemical engineer that works in a nearby plant where another explosion has just been reported. The HLS AA discovers that if the two smoke plumes intersect it will create a deadly acid mist. The AA plots the smoke plumes on the map and notifies the HLS analyst that the combined plume will reach a large stadium with an ongoing football game in 15 minutes. The AA immediately initiates a phone call between the HLS analyst and stadium security. In this scenario all the AAs can be played by CAST agents. The workflow among these AAs is a typical teamwork process. It can be shared so that the AAs could better anticipate and help each other's information needs, and make better decisions in such a highly dynamic situation.

Agent-aided collaborative decision-making has been investigated extensively in the literature [24][4][27]. The decision-making approach implemented in CAST is different from the existing work in several aspects. First, the decision-making is based on the common understanding of the team process and team progress. Such a shared mental model and the proactive information exchanges elicited from the shared mental model enable CAST agents to make better decisions in a dynamic environment. Second, CAST agents are able to make three kinds of decisions: (1) decide on the next course of actions when facing a choice point; (2) decide whether to inform others when obtaining a new piece of information; (3) decide whether to do information fusion when the recipient's cognitive capacity is considered. While STEAM [37]employed a theoretic approach similar to CAST in deciding on inter-agent communication, it does not allow agents to make the first and the third kinds of decisions. Third, our approach centers on proactive information gathering before making a decision. In [32] a collaborative decision-making framework is proposed to address the homeland security problem. However, it mainly focuses on the argumentation process where decision makers challenge each other when conflict occurs.

7. Conclusion

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 to addressing these challenges in areas ranging from digital force to collaborative intelligence analysis.

This paper focused on the cognitive construct "shared mental model", and explored how the SMM implemented in CAST can support teams of agents in their decision-making tasks. More specifically, we extended CAST with a decision-making module, which allows CAST agents to make decisions on the next course of actions when facing a choice point; on whether to inform others when obtaining a new piece of information; and on whether to fuse information before delivery when the recipient's cognitive capacity is considered. SMM is critical in making all of these kinds of decisions.

We evaluated the impacts of the SMM-supported decision-makings on the performance of agent teams through two experiments in a simulated NCW environment. The results of both experiments are positive, which indicates the implemented decision-making approaches can be effective in improving team performance. While this is still far from conclusive, from here we can gain some insights on the use of SMM in supporting decision-making teams composed of software agents; this provides us a stepstone for further exploring the SMM approach in supporting mixed human/agent teams in their collaborative problem solving.

Acknowledgements

This research was supported by AFOSR MURI grant No. F49620-00-1-0326, and Army Research Office grant No. DAAD19-01-1-0504.

References

[1] D. Alberts, J. Garstka, and F. Stein, Network Centric Warfare, CCRP Press (1999).

[2] E. Blickensderfer, J. A. Cannon-Bowers and E. Salas, Cross-Training and Team Performance, in J. A. Cannon-Bowers & E. Salas, Eds., Making Decisions Under Stress, Washington DC: American Psychological Association (1998) 299-311.

[3] R. J. Brachman and H.J. Levesque, The tractability of subsumption in frame-based description languages, Proceedings of the AAAI-84, Austin, Texas (1984) 34-37.

[4] T. Bui, and J. Lee. An agent-based framework for building decision support systems, Decision Support Systems 25 (1999) 225-237.

[5] K. Burns, Mental Models and Normal Errors, Proceedings of the 5th Conference on Naturalistic Decision Making, Tammsvik, Sweden (2000).

[6] 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 (1998) 92-101.

[7] 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 (1990) 1-4.

[8] J. A. Cannon-Bowers, E. Salas, and S.A. Converse, Shared mental models in expert team decision making, Individual and group decision making, Castellan, NJ (1993) 221-246.

[9] P. R. Cohen and H.J. Levesque, On team formation, in J. Hintikka, R. Tuomela, Eds., Contemporary Action Theory (1997).

[10] P. R. Cohen and H. J. Levesque, Teamwork, Nous 25(4) (1991) 487-512.

[11] M. D. Coovert and K. McNelis, Team Decision Making and Performance: A Review and Proposed Modeling Approach Employing Petri Nets, in R. W. Swezey, E. Salas, Eds., Teams: Their Training and Performance, Ablex Pub Corp (1992).

[12] D. Fensel, F.V. Harmelen, I. horrocks, D.L. McGuinness, and P.F. Patel-Schneider, Oil: An Ontology Infrastructure for the Semantic Web, IEEE Intelligent Systems (2001) 38-45.

[13] J.A. Giampapa and K. Sycara, Team-Oriented Agent Coordination in the RETSINA Multi-Agent System, tech. report CMU-RI-TR-02-34, Robotics Institute, Carnegie Mellon University (Dec. 2002).

[14] A. Gomez-Perez, and O. Corcho, Ontology Languages for the Semantic Web, IEEE Intelligent Systems 17(1) (2002) 54-59.

[15] B. Grosz and S. Kraus, Collaborative plans for complex group actions, Artificial Intelligence (1996) 269-358.

[16] B. Grosz and S. Kraus, The Evolution of SharedPlans, in A. Rao, M. Wooldridge, Eds., Foundations and Theories of Rational Agencies (1998) 227-262.

[17] B. Grosz and C. Sidner, Plans for discourse, in P. Cohen, J. Morgan, M. Pollack, Eds., Intentions in communication, MIT Press (1990) 417-444.

[18] G. Halford, W. Wilson, and W. Phillips, Processing capacity defined by relational complexity: Implications for comparative, developmental and cognitive psychology, Behavioral Brain Sciences 21(6) (1998) 803-831.

[19] J.R. Hollenbeck, D.J. Sego, D.R. Ilgen, D.A. Major, J. Hedlund, J. Phillips, Team Decision-making accuracy under difficult conditions: construct validation of potential manipulations using the TIDE2 simulation, in M.T. Brannick, E. Salas, C. Prince, Eds., Team Performance Assessment and Measurement, Lawrence Erlbaum Asso. Publishers (1997).

[20] 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 (1988) 247-302.

[21] D. R. Ilgen, D.A. Major, J.R. Hollenbeck, and D.J. Sego, Team Research in the 1990's, in M. Chemers, R. A. Ayman, Eds., Leadership Theory and Research: Perspectives and Directions, San Diego: Academic Press (1993).

[22] T.R. Ioerger, JARE: Java Automated Reasoning Engine, http://jare.sourceforge.net.

[23] N.R. Jennings, Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems using Joint Intentions, Artificial Intelligence 75(2) (1995) 195-240.

[24] V. Karan, D. Kerr, U. Murthy, and A. Vinze, Information technology support for collaborative decision making in auditing: An experimental investigation, Decision Support Systems 16 (1996) 81-194.

[25] G.A. Klein, Recognition-primed decision making, Sources of power: How people make decisions (1998) 15-30.

[26] P. Kogut, J. Yen, Y. Leung, S. Sun, R. Wang, T. Mielczarek, B. Hellar, Proactive Information Gathering for Homeland Security Teams, Communication of ACM (2004).

[27] F. Maturana and D. Norrie, Distributed decision-making using the contract net within a mediator architecture, Decision Support Systems 20 (1997) 53-64.

[28] National Strategy for Homeland Security, US Office of Homeland Security, www.whitehouse.gov/homeland/book/ (July 2002).

[29] J. Orasanu, Shared Mental Models and Crew Performance, in Proceedings of the 34 Annual Meeting of the Human Factors Society, Orlando, FL (1990).

[30] M.E. Pollack, Plans as complex mental attitudes, in P. Cohen, J. Morgan, M. Pollack, Eds., Intentions in communication, Bradford Books, MIT Press (1990).

[31] C. Porter, J.R. Hollenbeck, D.R. Ilgen, A.P.J. Ellis, B.J. West, and H.K. 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 (2003).

[32] T. S. Raghu, R. Ramesh, Andrew B. Whinston, Addressing the Homeland Security Problem: A Collaborative Decision-Making Framework, in H. Chen, et al., Eds., Intelligence and Security Informatics, LNCS-2665, Springer (2003) 249-265.

[33] J.R. Rentsch and R.J. Hall, Members of great teams think alike: A model of team effectiveness and schema similarity among team members, in M. M. Beyerlein, D. A. Johnson, Eds., Advances in interdisciplinary studies of work teams 1, Series on self-managed work teams, Greenwich, CT: JAI Press (1994) 223-262.

[34] C. Rich and C. Sidner, COLLAGEN: When agents collaborate with people, Proceedings of the International Conference on Autonomous Agents (1997).

[35] W.B. Rouse, J. A. Cannon-Bowers, and E. Salas, The role of mental models in team performance in complex systems, IEEE Transactions on Systems, Man, and Cybernetics 22(6) (1992) 1296-1308.

[36] K. Sycara, K. Decker, A. Pannu, M. Williamson, and D. Zeng, Distributed Intelligent agents, IEEE Expert, Intelligent Systems and their Applications 11(6) (1996) 36-45.

[37] M. Tambe, Towards flexible teamwork, Journal of Artificial Intelligence Research 7 (1997) 83-124.

[38] G. Tidhar, C. Heinze, and M. Selvestrel, Flying Together: Modeling Air Mission Teams, Journal of Applied Intelligence 1(1) (1998).

[39] 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 (1996) 87-100.

[40] D. Xu, R. Volz, T. Ioerger, J. Yen, Modeling and Analyzing Multi-Agent Behaviors Using Predicate/Transition Nets, International Journal of Software Engineering and Knowledge Engineering 13(1) (2003) 103-124.

[41] J. Yen, X. Fan, and R.A. Volz, On Proactive Delivery of Needed Information to Teammates, in Proceedings of the AAMAS 2002 Workshop of Teamwork and Coalition Formation, Italy, (2002) 53-61.

[42] J. Yen, X. Fan, and R.A. Volz, Proactive Communications in Agent Teamwork, in Advances in Agent Communication (LNAI-2922), Springer (2004) 271-290.

[43] J. Yen, J. Yin, T.R. Ioerger, M. Miller, D. Xu, and R.A. Volz, CAST: Collaborative Agents for Simulating Teamwork, in Proceedings of the 7th International Joint Conference on Artificial Intelligence (2001) 1135-1142.

[44] J. Yin, M. 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, Barcelona, Spain (2000) 427-434.

[45] D. Zeng, H. Chen, D. Daspit, F. Shan, S. Nandiraju, M. Chau, C. Lin, COPLINK Agent: An Architecture for Information Monitoring and Sharing in Law Enforcement, in H. Chen, et al., Eds., Intelligence and Security Informatics, LNCS-2665, Springer (2003) 281-295.

[46] J. Zhao, H. Bi, H. Chen, Collaborative Workflow Management for Interagency Crime Analysis, in H. Chen, et al., Eds., Intelligence and Security Informatics, LNCS-2665, Springer (2003) 266-280.

John Yen is University Professor of Information Sciences and Technology and Professor-in-Charge at the School of Information Sciences and Technology at The Pennsylvania State University. He is the founding director of the Intelligent Agent Laboratory at Penn State. He holds a PhD in computer science from University of California, Berkeley. He is a Fellow of IEEE. His current research focus is developing team-based agent technologies for supporting human decision making teams as well as transferring these technologies into applications related to homeland security, information fusion, and web service composition.

Xiaocong Fan received the PhD degree from Nanjing University in 1999. From 2000-2002, he worked at the Turku Centre for Computer Science and the Computer Science Department of Abo Akademi University in Finland, where he participated in the projects SOCOS and SPROUT, which developed a methodology for software platform construction based on the Refinement Calculus. In 2002, he moved to the Pennsylvania State University and joined the Intelligent Agent Lab as a senior researcher. He currently works on formal agent theories in teamwork, and applications using these theories.

Shuang Sun is a PhD candidate in School of Information Sciences and Technology (IST) in the Pennsylvania State University. His research interests include team-based intelligent agent technology and e-commerce. Before he joined IST in 2001, he was a business application consultant at NOKIA, leading and implementing SAP projects. Previous to that, Shuang worked as a consultant at Deloitte & Touche. He provided information technology solutions to address clients' requirements on finance and accounting information management. Shuang is a certified professional on products of many leading IT organizations including SAP,Microsoft, Lotus, and Nortel.

Timothy Hanratty - is a senior computer scientist at the US Army Research Laboratory. His professional interest focus on applied research and development in the areas of knowledge engineering and advance decision and collaborative technologies. Hanratty received his BS in computer science from Towson University and his MS from Johns Hopkins University.

John Dumer - is a senior computer scientist at the US Army Research Laboratory. His principal research areas include fuzzy logic, reasoning under uncertainty and cooperative expert system technologies. Dumer holds a BS in computer science/communications for Towson University.

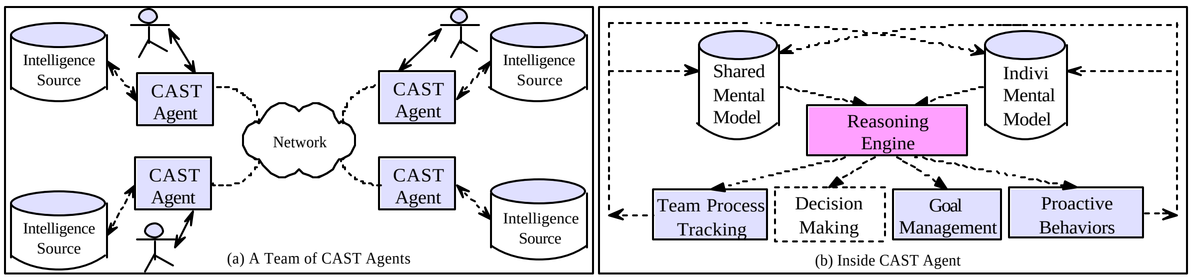

Fig.1 The CAST architecture

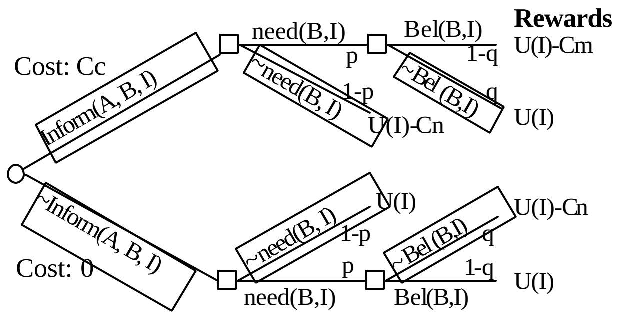

Fig.2 Decision tree for proactive communication

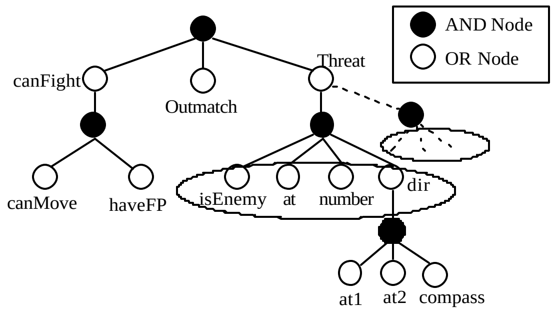

Fig.3 Levels of Information Needs

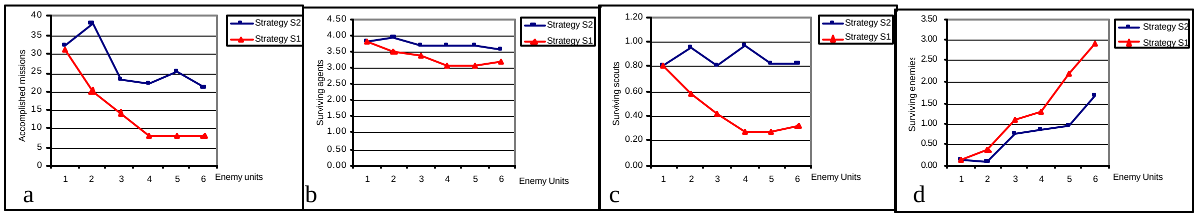

Fig.4 Results of experiment I

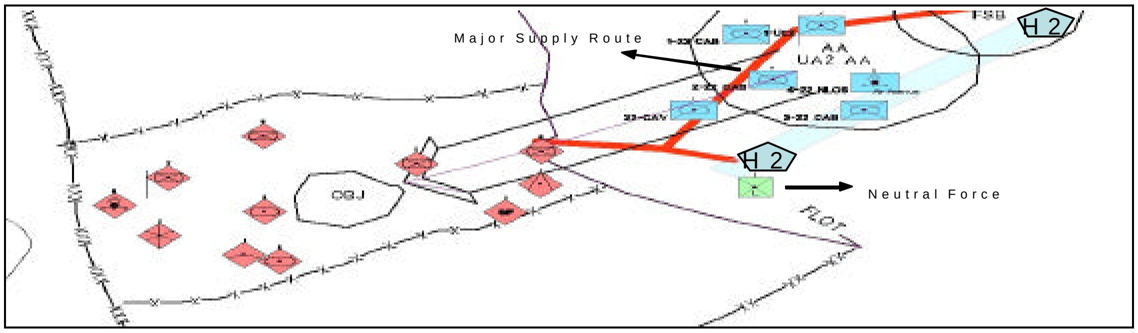

Fig. 5 The Sample Military Scenario Map

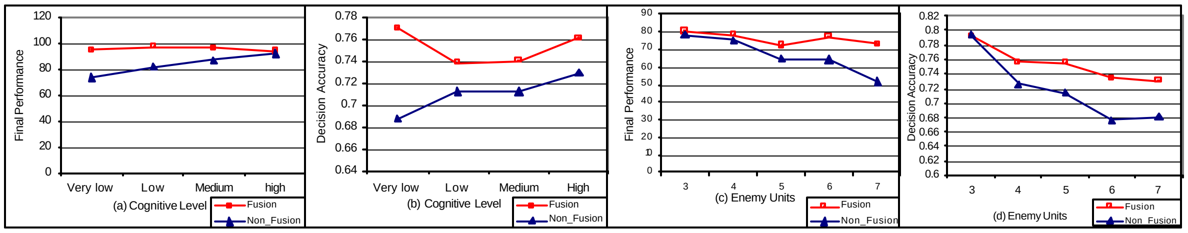

Fig.6 Results of experiment II

Table 1. Example Inference knowledge in CAST

(1)	Threat(?e,?loc,?dir,?num)←IsEnemy(?e), At(?e,?loc,NOW),Dir(?e,?dir),Number(?e,?num)
(2)	Dir(?e,?dir)←At(?e,?l1,NOW-1), At(?e,?l2,NOW),Compass(?l1,?l2,?dir)
(3)	CanFight(Self)←HaveFP(Self),CanMove(Self)