Extending the Recognition-Primed Decision Model to Support Human-Agent Collaboration

Xiaocong Fan, Shuang Sun, Michale McNeese, and John Yen
School of Information Sciences and Technology
The Pennsylvania State University
University Park, PA 16802
{zfan, ssun, mmcneese, jyen}@ist.psu.edu

ABSTRACT

There has been much research investigating team cognition, naturalistic decision making, and collaborative technology as it relates to real world, complex domains of practice. However, there has been limited work in incorporating naturalistic decision making models for supporting distributed team decision making. The aim of this research is to support human decision making teams using cognitive agents empowered by a collaborative Recognition-Primed Decision model. In this paper, we first describe an RPD-enabled agent architecture (R-CAST), in which we have implemented an internal mechanism of decision-making adaptation based on collaborative expectancy monitoring, and an information exchange mechanism driven by relevant cue analysis. We have evaluated R-CAST agents in a real-time simulation environment, feeding teams with frequent decision-making tasks under different tempo situations. While the result conforms to psychological findings that human team members are extremely sensitive to their workload in high-tempo situations, it clearly indicates that human teams, when supported by R-CAST agents, can perform better in the sense that they can maintain team performance at acceptable levels in high time pressure situations.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: General—*Cognitive simulation*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents, Multiagent systems*

General Terms

Design, Experimentation, Human Factors

Keywords

Human-centered teamwork, Human-agent collaboration, Naturalistic decision making, Recognition refinement, Shared situation awareness

1. INTRODUCTION

Human teamwork in dynamic, uncertain environments is often threatened by the tension between information overload and distributed cognition. On the one hand, to balance the cognitive demands on team members, the data, information, and knowledge are distributed across people, objects, tools, and environments [3]. But on the other hand, the team needs to develop a shared understanding of the current situation in order to make better and faster decisions. For example, to enable early detection and successful processing of potential terrorist threats, team members must effectively work together to quickly process and fuse information from multiple sources. This necessitates information exchange, which has to be done cautiously to avoid the information overload problem [17].

There has been much theory and research presented that investigates team cognition, naturalistic decision making, and collaborative technology [10] as it relates to real world, complex domains of practice. For instance, one theory that attempts to understand the dynamics of team cognition introduces the concept of a "shared mental model" [1], which refers to an overlapping understanding among members of the team regarding their objectives, structure, process, etc. However, there has been very little work in looking at cognitive agent architectures as a means to support distributed team cognition and decision making. This is particularly true as applied to naturalistic decision making theories, one of which is Klein's Recognition-Primed Decision framework (RPD) [4, 5], which starts to capture how human experts make decisions based on the recognition of past experiences that are similar to the current situation.

This research supports human teams to make faster and better decisions using cognitive agents that encourage both agent-agent collaboration and human-agent collaboration. In this investigation, the RPD model is chosen for two reasons. First, RPD offers a well-structured process for better solving ill-structured problems where there is no time for extensive reasoning. Teamwide collaboration opportunities can be naturally embedded into the RPD process; this enables us to further investigate dynamic information sharing problems and distributed team cognition problems [3]. Second, RPD focuses on recognizing the similarity between the current decision situation and previous decision experiences, which is claimed as the way how experts usually make decisions in complex situations. Implementing agents with a computational RPD model can encourage close *agent-human collaboration* in the decision-making process (e.g.,

adjustable autonomy); this advocates the view of human-centered teamwork [15], where from a human's perspective, agents are not just black-boxes providing decision making supports, but rather active peers that humans can directly interact with.

While Norling, Sonenberg, and Ronnquist [12] and Norling [11] have examined the integration of RPD model into BDI agent framework, they mainly focused on using reinforcement learning to enable agents to recognize the subtle cues that distinguish one situation from another. Many of other issues have been ignored, such as agent-agent collaboration and human-agent collaboration in the decision making process, adaptation of decision strategies and coordination strategies under time pressure, and effective team cognition in complex, distributed environments. Psychological studies (e.g., [8]) have shown that team members are extremely sensitive to their workload in high-tempo situations. In this paper, we focus on understanding how RPD-enabled agents may influence the performance of human teams in high-tempo situations.

The remainder of this paper is organized as follows. In Section 2 we give some backgrounds about the RPD model, the adaptive team model, and DDD– a simulation tool for distributed dynamic decision making. In Section 3, we introduce R-CAST—an agent architecture implementing a collaborative RPD model, focusing on experience organization and the *iterative*, *collaborative* RPD process. In Section 4, we report the evaluation results of the adaptive decision-making strategy implemented in R-CAST, and report the experimental results of the performance of pure human teams and human teams supported by R-CAST agents. Section 5 concludes the paper.

2. BACKGROUND

The RPD Model The RPD model [4] includes a recognition phase and an evaluation phase. In the recognition phase, a strategy called "feature-matching" is firstly used to develop situation awareness. Typically, a decision maker synthesizes the observed features about the current decision situation into appropriate cues or pattern of cues, then tries to recall similar cases by matching the synthesized cues with previous experience. In the case that feature-matching cannot provide an acceptable solution due to lack of information or experience, "story-building" is used to develop a potential explanation of how the current situation might have been emerging, and a workable solution can be suggested afterward. The recognition phase has four products: relevant cues (what to pay attention to), plausible goals (which goals make sense), expectancy (what will happen next), and courses of action (what actions worked before in this type of situation). Due to the dynamic and uncertain nature of the environment, it is very important to monitor the status of expectancy because a decision maker may have misinterpreted the current situation but he/she cannot recognize it until some expectancy is invalidated as the situation evolves further. In such cases, the decision maker may need to further diagnose the current situation (e.g., to gather more information).

In the evaluation phase, a decision maker evaluates a course of action by imaging how it will evolve. In case that a course of action does not work for the current situation, the decision maker can either adjust it, or reject it and examine another option until a workable solution is obtained.
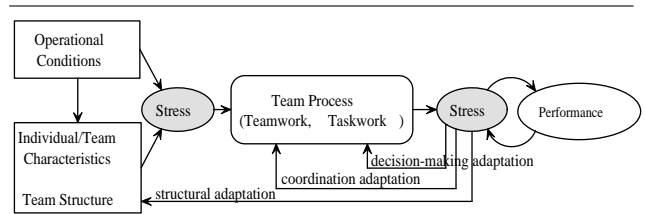

Figure 1: The Adaptive Team Model [12]

Adaptive Team Model Successful teams are able to use their knowledge of the situation being investigated to adapt their behaviors to dynamic demands. Figure 1 illustrates the Adaptive Team Model proposed by Serfaty, Entin, and Johnston [14]. At the input level, stress is modeled as the effects of operational conditions on team members and the team structure. At the output level, stress is produced as a function of the type of taskwork and teamwork strategies used by the teams. The central premise of the Adaptive Team Model is that high-performing teams are able to cope with stress through decision strategy adaptation, coordination strategy adaptation, and structural adaptation, in an effort to maintain acceptable team performance while keeping the perceived stress at tolerable levels. Drawing upon this theory, the R-CAST agent architecture to be introduced in Section 3 supports dynamic adaptation in making decisions under time stress situations.

DDD DDD (Distributed Dynamic Decisionmaking) is a proven real-time simulation environment for measuring team performance [6, 9]. It is rooted in the Integrated Team Model [13] and the Adaptive Team Model [14]. DDD was designed to capture the essential elements of different command and control team tasks. It allows researchers to generate various scenarios by manipulating task structures (varying the number, type, uncertainty, and timing of tasks to be processed) and team organizational structures (changing authority, team size, the ownership and control of assets, the availability of information, types of communication, etc.). By controlling the assets owned by them, experiment participants can coordinate to accomplish tasks, plan actions to gather information, and share information and resources. DDD has been successfully used in understanding decision making under stress [14], and in studying the performance of AWACS Weapons Directors teams, Search and Rescue teams, and Joint Task Force teams. In our experiments, DDD was used as the test bed for comparing the performance of human teams and agent-supported human teams under varying time stress situations.

3. R-CAST: RPD-ENABLED COLLABORATIVE AGENTS

RPD is a human-like decision-making model. We extend the RPD framework to build a computational model for supporting team decision making, providing the capabilities of capturing human experiences, supporting both human-agent and agent-agent collaboration, and making decisions adaptive to the situation changes. In this section, we describe the R-CAST agent architecture, the representation of experience, and the collaborative decision making process.
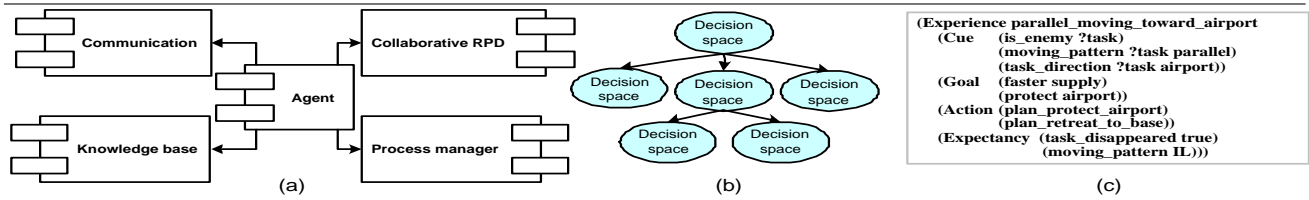
Figure 2: (a) R-CAST agent architecture, (b) Experience organization, (c) A sample experience.

3.1 The R-CAST agent Architecture

The R-CAST agent architecture—an extention of the CAST architecture [19, 18]–is composed of a communication manager, a knowledge base, a process manager, and a collaborative RPD module (see Figure 2(a)).

- The *communication manager* governs inter-agent communication. An agent may either initiate a new conversation context or simply follow existing ones. The manager organizes related messages into task-indexed conversation sessions, and monitors the development of on-going conversation protocols.

- The *knowledge base* is a forward-chaining rule-based system. Each R-CAST agent has an internal knowledge base (KB) to maintain what it believes regarding the external world and the other agents. The knowledge base engine is active, proof-preserving, and time-sensitive. It is active in the sense that it is able to reason about missing information relative to the current tasks, and proactively find ways to satisfy the inferred information requirements. It is proof-preserving in the sense that the proof-trace (i.e., supporting facts) of a query is preserved; this is important when information link-analysis is needed, and can be very useful in planning for information gathering in subsequent activities. It is time-sensitive in the sense that it can "forget" certain information as time proceeds. Different types of information may vary in their persistence durations, which can be pre-specified according to the nature of an information type.

- The *process manager* manages the templates of predefined plans, each of which contains preconditions, termination conditions, effects, and a process body. Upon being requested by the decision-making module, the process manager can instantiate plan instances from appropriate templates. An agent may run multiple plan instances simultaneously, each of which can be in an active, suspended, or terminated state. The process manager is responsible for scheduling the execution of plan instances based on the constraints associated with the instances and the KB's current state.

- The *collaborative RPD module* implements the RPD process and allows team members (both humans and other R-CAST agents) to take opportunities to collaborate with each other during the RPD process.

3.2 Experience Organization

Decision making in the RPD model is based on experiences. Therefore, it is a critical design question to find appropriate ways to represent and organize experiences acquired through cognitive task analysis from domain experts.

Typically, human decision makers are organized in authority hierarchies, and a higher-level decision-making task usually depends on the results of lower-level decision-making tasks. It is thus natural to enable R-CAST agents to support hierarchical multiple-level decision makings. This is powerful in two ways: (1) decision making and information processing can be tightly coupled at multiple levels, if we view decision making as a process of fusing input information into decision results; (2) decision-making knowledge and expertise are often distributed among team members. The hierarchical approach encourages team-level collaboration at multiple levels within the same decision context.

Each decision-making task has certain knowledge (expertise) requirements on the decision maker. Two decision-making tasks belong to the same *type* if they place the same knowledge requirements (e.g., the same collection of cues to consider). We thus use the concept "decision space" to organize experiences for complex domain problems in a hierarchical way (see Figure 2(b)), where experiences related to one decision type are maintained in one experience knowledge base (EKB). Upon getting (or identifying) a decision-making task, an agent will make it mutually known among the team. Those agents who have the required decision-making knowledge can proactively help on the task or decision tasks at lower levels. Those agents who do not have the required knowledge (they thus cannot easily anticipate the decision maker's needs by themselves) can also provide help if they receive information requests from the decision maker or other teammates.

Each experience has four parts: cues, goals, course of actions, and expectancies. In our system, cues, goals, and expectancies of an experience are all represented as predicates. To facilitate the agent kernel to do feature matching, dynamic situations are also internally represented as a collection of predicates in KB. Formally, an experience is denoted as $e_i = \langle C_i, G_i, E_i, A_i \rangle$, where C_i, G_i, and E_i are collections of predicates, and A_i is a set of plan names, referring to pre-defined courses of actions. Figure 2(c) gives an example experience for dealing with enemies moving parallely toward an airport. This experience can be used to protect the airport and ensure faster supply delivery. To choose this experience, the agent has to gather enough information to determine whether a task is an enemy unit, whether its moving pattern is parallel, and whether the task is moving toward the airport. If the current situation matches this experience, the agent has to monitor the associated expectancies. In case that the enemies disappear or the moving pattern is changed, the agent has to make a new decision, adapting to the situation changes.
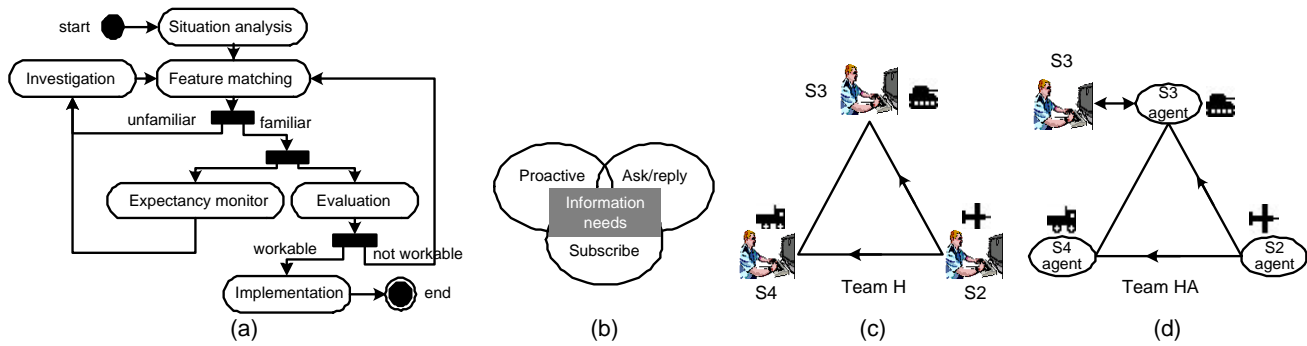
947

Figure 3: (a) The computational RPD process, (b) three communication modes driven by information needs, (c) the structure of human teams, (d) the structure of agent-supported human teams.

3.3 The Collaborative RPD Process

R-CAST has realized a set of functions corresponding to the main steps of the RPD model: situation analysis, recognition, evaluation, implementation, and expectancy monitoring. Figure 3(a) shows the implemented RPD process.

3.3.1 Situation Analysis

In responding to a decision task, an R-CAST agent first explores the decision space hierarchy to choose one that is most applicable to the decision task by comparing the current situation with the cues considered by a decision space. In case that this R-CAST agent cannot handle the task due to lack of expertise, it will try to find a competent R-CAST agent and transfer the task to that teammate. The agent whoever makes a commitment to a decision task will also let others know so that they can help in the rest of the decision making process.

3.3.2 Recognition

The recognition phase involves two sub-processes: investigation and feature matching.

Investigation is the process for collecting missing information; it is the key to evolving recognitions. Given a task, if an agent has the capacity and capability to gather information, it can activate an information-seeking plan. For example, an agent can launch a helicopter to scout enemy information. If the agent is incapable or has insufficient resources to do so, it can send an information request to others. In addition, other agents who have anticipated the decision maker's information needs may also proactively provide relevant information to the decision maker.

Feature matching While an R-CAST agent is gathering information, it also triggers the feature matching function to check whether there are past experiences similar to the current situation. Because the information regarding the current situation is recorded in the agent's KB, the feature matching process simply iterates over the experiences in the active EKB and casts queries to the agent's KB with the cues to be evaluated. The experiences with the most number of cues satisfied with respect to the KB are the recognition results.

3.3.3 Evaluation and Execution

Evaluation is a process for selecting a workable course of action. For human decision makers, evaluation is a mental simulation process: people imagine how the course of ac-

tion may evolve and judge whether the relevant goals can be achieved. For an R-CAST agent, we simplify the process by simulating the effects of the selected plan (i.e., the chosen course of action) in the knowledge base (KB). First, the RPD-agent checks whether the preconditions of the selected plan is satisfiable with respect to its KB. If so, the agent asserts the effects of the plan into its KB and uses the KB engine to check whether the relevant goals become true. If a plan can pass such a two-phase KB evaluation, it is deemed as a workable solution for the current situation and the agent will coordinate with other teammates to execute the plan. Otherwise, the agent has to make another round of recognition, going through the RPD process again.

3.3.4 Expectancy Monitoring

After an R-CAST agent makes a recognition, it will continuously monitor the associated expectancies until the completion of the selected course of action. Expectancy monitoring is one of the key features implemented in R-CAST to support adaptive decision making [14]. First, expectancies can be used to initiate a complete new decision. An expectancy states what will happen, serving as a gate-condition for keeping following the current recognition. Some expectancies may be so *crucial* that whenever they conflict with the new observed facts, it indicates the decision maker has heavily misinterpreted the current situation. In such cases, the R-CAST agent has to diagnose the current recognition, re-considering the *whole* space of the active EKB for another round.

Second, R-CAST agents can use expectancies to *refine* a decision, leveraging some structures within the active EKB. The invalidation of some expectancies may indicate that the once workable recognition is no longer applicable to the changing situation. The already executed part of the selected course of actions may still make sense, but the rest has to be adjusted. In such cases, the R-CAST agent can start another round of recognition, but it is sufficient to only consider the additional cues that have certain relations with the expectancies being invalidated. Let the experiences selected in iteration i and $i+1$ be $e_i = \langle C_i, G_i, E_i, A_i \rangle$ and $e_{i+1} = \langle C_{i+1}, G_{i+1}, E_{i+1}, A_{i+1} \rangle$, respectively, and let $\sqsubseteq$ be the recognition (experience) refinement relation. Then, if $e_i \sqsubseteq e_{i+1}$, what exactly are the relations that exist between the corresponding components of e_i and e_{i+1}? After reviewing the experiences used in our experiments, we did find that $C_i \subseteq C_{i+1}$, $G_i = G_{i+1}$, and $A_i \prec A_{i+1}$ ($\prec$ is a sequence pre-

fix relation) hold for most cases. Moreover, it is the invalidated part of E_i that contributes to the difference between C_i and C_{i+1}. From such a perspective, experiences in an EKB can be viewed as being partitioned by some experience refinement relation. The elicitation of experience refinement relations and how to classify crucial/non-crucial expectancies are important for better understanding the structure inside a decision space. However, detailed discussion of these issues is out of the scope of this paper.

Therefore, we have implemented an *iterative*, computational RPD model in R-CAST, which explicitly incorporates the idea of "recognition refinement", and supports situation reconsideration during action execution phase. The computational model is also much more flexible for us to investigate the time pressure issue. Since R-CAST agents can make a sequence of decisions (the length of the sequence is restricted by the external time pressure), with one decision refining the preceding ones, it can always return an acceptable decision relative to the timing constraints. This virtually guarantees no critical decision is missed under stress situations. Figure 4 is the pseudo-code of the iterative algorithm for RPD.

IterativeRPD(KB,EKB)
/* e_n is of form $\langle C_n, G_n, E_n, A_n \rangle$. EC/EN stores
  crucial/non-crucial expectancies being invalidated. */
1. SituationAnalysis();
2. CS = GetCurrentSituation(KB);
3. e_0 = Recognition(CS,EKB);
4. InformRecognition(e_0);
5. For (i=0; ; i++)
6.   [EC,EN] = validating(E_i);
7.   If (timeout is true)
8.   goto 1; /*prepare for the next task*/
9.   CS = GetCurrentSituation(KB);
10.   If (EC is not null)
11. e_{i+1} = Recognition(CS,EKB);
12.   InformRecognition(e_{i+1});
13.   Else If (EN is not null)
14. e_{i+1} = RefiningRecognition(CS,EKB);
15.   InformRecognition(e_{i+1})

Figure 4: The iterative algorithm for RPD.

3.3.5 *Collaboration Opportunities*

The computational RPD model implemented in R-CAST is also a *collaborative* model. As shown in Figure 3(d), each R-CAST agent (e.g., S3 agent) may have a human partner, who can override the decisions suggested by the R-CAST agent. An R-CAST agent can also team up with other R-CAST agents and interact with them. As far as a specific decision task is concerned, if one RPD-agent who has the required expertise is designated as the final decision maker, all the other R-CAST agents will be *supporters* (of course, the supporters may be the final decision makers for other tasks). In the current implementation, a decision maker agent can collaborate with its human partner and the other supporting agents along the RPD process. Here we mainly focus on exchanging information relevant to establishing situation awareness for decision makings.

A decision maker agent can derive its information needs regarding the current decision task from the cues considered in the active experience base (EKB) and the expectancies of the experiences that are found similar to the current situation. As shown in Figure 3(b), the derived information needs can be satisfied in three ways. First, teammates can proactively provide the decision maker agent (DMA) with information relevant to the cues that the DMA is considering. As we mentioned earlier, this happens when a teammate has been informed of the decision task, and the teammate has the required expertise (e.g., its experiences overlap with the DMA's). Second, in cases where agents do not have overlapped experiences, the decision maker may explicitly request information from teammates. Here, 'ask-reply' becomes a handy way to compensate the limitations of proactive communication. Third, the DMA can subscribe information relevant to the expectancies that need to be continuously monitored. When the DMA informs other teammates about the recognition, it is also implicitly requesting them to monitor the expectancies. Such collaborative expectancy monitoring takes full advantage of the team's distributed cognition, so that the DMA can terminate the activity resulted from a wrong recognition at the earliest opportunity.

4. EXPERIMENTS

We describe the design of the scenarios used in the experiments, evaluate the adaptive decision making feature of R-CAST, and then report the result of the experiments for understanding R-CAST agents co-working with humans under time stress.

4.1 Scenario Design

The basic scenario involves a blue (friendly) force consisting of three battle functional areas (BFAs): the intelligence cell (S2), the operations cell (S3), and the logistics cell (S4). In the battlefield, there is a supply route connecting an airport A (the bold square in Figure 5(a)) and a target area T (the bold ring in Figure 5(a)) in the frontier. The overall goal of the blue force is to protect the airport A and the target area T , and to ensure as many rounds of supplies as possible are delivered by S4 from A to T . For the purposes of our study the capabilities of the BFAs have been simplified and defined as follows:

- S2, who has an unmanned aerial vehicle (UAV) under its control, is able to collect information regarding the approaching objects (tasks), and identify whether a task is a neutral force or enemy unit.

- S3, who has a tank under its control, is able to destroy enemies and protect the supply route. The tank will be heavily penalized if it attacks an unidentified task.

- S4, who has one truck under its control, is able to deliver supplies from A to T along the supply route. If within the attack range of an approaching enemy, the truck will be destroyed.

Clearly, the BFAs have to collaborate with each other to have a better team performance. Both S3 and S4 rely on S2 to share information regarding the approaching tasks. Otherwise, the airport or the target area might get attacked, and the truck has to move away to avoid being attacked (in such a case, it wastes time unnecessarily if the task is actually a neutral force). S4 also relies on S3 to protect the truck to ensure faster delivery.

Figure 5: (a) A screen shot of S2 cell, (b) The interface of R-CAST agent: a human uses the left panel to input the attack patterns.

We used DDD as the test bed. DDD allows us to design scenarios with different classes of tasks with different characteristics. We defined 4 task classes (2 classes of neutral force and 2 classes of enemy units) with different strength values (the minimum resources required to attack a task) and different maximum moving speeds. Each task class can generate many instances. Two task instances of any type combination form one joint-task to the blue force. There is a 2-second interval in between two joint-tasks. We also designed 6 moving patterns for the task instances:

H: one task moves toward A , the other toward T ;
V: both tasks move toward A ;
K: move like V initially, then one moves away;
IL: move like H initially, then the one toward A moves away;
JI: move like H initially, then the one toward T moves away;
JL: move like H initially, then both move away.

When a task moves away, even if it is an enemy unit, it has no threat to the blue force, which can therefore save resources for the next round of attack. Since the H pattern may change to an IL, JI, or JL pattern, and the V pattern may change to a K pattern, the decision makers have to be alert to adapt their decisions in a timely manner. By varying the moving patterns of attack, together with task type combinations, we were able to control task complexity in our experiments.

We did some testing experiments first. The participants were asked to play the roles of BFAs using some testing scenarios. Useful experiences were elicited from the testing experiments (see Figure 2(c) for an example). These experiences were used by the R-CAST agents in the following experiments.

## 4.2 Evaluating The Adaptive Decision Making

In the first experiment, two human participants need to play the role of S2 and S4, respectively. An R-CAST agent,

together with a human partner, plays the role of S3. The human partner of S3 only informs the R-CAST agent of the task moving patterns by clicking the corresponding buttons in the human-agent interface (Figure 5(b)). We designed a scenario with 48 joint-tasks, one pops up after the preceding one has been handled. Most of the joint-tasks in the scenario are K, IL, JL, and JI patterns. We observe whether the tank controlled by the R-CAST agent can adapt to pattern changes. The result shows that, for all the joint-attacks with pattern changes, if the human can inform the changes in time, the R-CAST agent can terminate the current course of action and make another decision according to the changed situation. Take one case as an example:

—: A joint-task of pattern V appeared;
—: S3 partner informed S3 agent by clicking the V-pattern button on the interface;
—: S2 identified one enemy unit and informed this to S3;
—: The S3 agent moved the tank to an attack point;
—: In the course of tank's moving, the enemy unit was moving away (the attack pattern changed to K);
—: S3 partner informed S3 agent by clicking the K-pattern button on the interface;
—: S3 agent terminated the tank's attacking plan.

4.3 Understanding R-CAST Under Stress

The following set of experiments is to understand how R-CAST agents can act as cognitive aids to human partners under stress situations. According to the Adaptive Team Model [14], we modeled time pressure as the effects of operational conditions. As we explained, the maximum moving speed can be specified for a task class. DDD also allows scenario designers to set the actual task moving speeds relative to the maximum speed. We thus designed 9 scenarios (each has a fixed task moving speed), with the speed ranging from 0.2 to 1, relative to the maximum speed. The faster the tasks move, the more stress on the experiment participants.
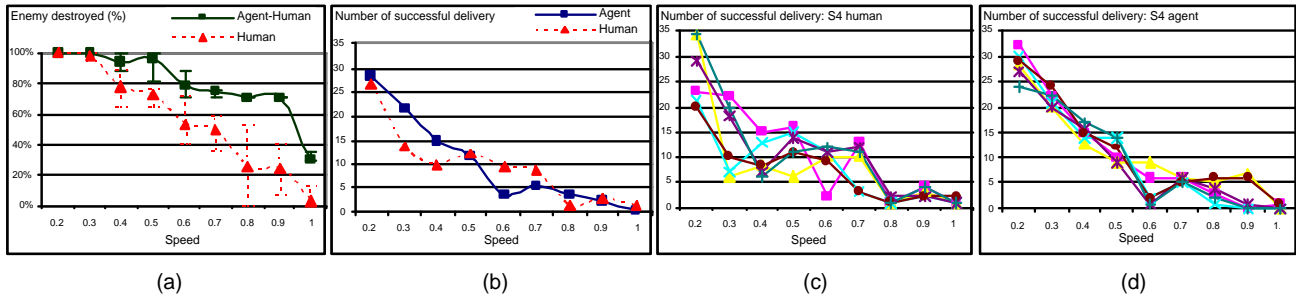
Figure 6: As stress increases, (a) enemies destroyed, (b) supplies delivered, (c) the performance of S4 played by humans, (d) the performance of S4 played by agents.

Each scenario contains 48 joint-tasks, with the attack patterns randomly configured. Depending on the task speeds, it takes 4 to 23 minutes to run a scenario.

To conduct our study, we designed two teams. Fig. 3(c) shows the structure of the human team (H), where each experiment participant has a DDD screen like Fig. 5(a). S2 is responsible for identifying enemy units for the whole team. S2 can (1) move the UAV to an approaching task; (2) identify a task by opening the Identify-Task window; and (3) transfer the enemy information to S3 and S4 through a Transfer-Info popup window. On S3 and S4's screens, the icon of a task is changed to either 'neutral' or 'enemy' only after S2 has transferred the task identification to them. S3 is responsible for attacking enemy units. S3 can (1) move the tank to an attacking point; (2) attack an enemy unit; and (3) move the tank to an optimal waiting point. S4 is responsible for delivering supplies. S4 can (1) move the truck to a target; and (2) move the truck away from a threat and wait until the threat disappears.

Fig. 3(d) shows the structure of human-agent team (HA), where two R-CAST agents play the roles of S2 and S4, respectively. One R-CAST agent, together with a human partner, plays the role of S3. While practically the human partner can override the S3-agent's decisions, in the experiments we only allow him/her to inform S3-agent of the attack patterns (because generally, humans are superior in spatial reasoning than agents). Similar to Team H, the S2-agent needs to transfer enemy information to the other agents. Clearly, Team HA requires both human-agent and agent-agent collaborations.

Our experiments involved six H teams and six HA teams, where human participants were selected from graduate students at PSU. Each team was tested with the 9 scenarios in a random sequence. The result is given in Fig.6.

Fig. 6(a) plots the average performance of the six HA teams and the six H teams in terms of what percent of enemy units were destroyed, and Fig. 6(b) plots their average performance in terms of how many rounds of supplies were successfully delivered. This clearly indicates that human-agent teams performed much better than pure human teams ($p \leq 0.001$ in terms of S3's performance, $p = 0.211$ in terms of S4's performance). When time pressure is low, both the HA and H teams could identify and destroy all the enemies. As time pressure increased, the performance difference of the HA teams and H teams also increased: the performance of H teams decreased dramatically when enemies attacked at

higher speeds, while the performance of HA teams were relatively stable. Even in the extreme case where tasks moved at the maximum speed (the last point in Fig. 6(a)), the mission was almost impossible for H teams, while the HA teams could still destroy 36% of the enemy units. While this result reinforces the findings that people are extremely sensitive to time pressure [8], it also indicates that as a cognitive aid, RPD-agents can enhance humans' performance under time pressure.

Fig. 6(a) also shows the deviation interval of each data point. It reveals that the performances of the six HA teams are mutually supportable, while the performances of the six H teams vary from each other dramatically. To have a better insight on how human personalities may affect the performance, Fig. 6(c) plots the performances of the S4s in the six H teams, and Fig. 6(d) plots the performances of the S4-agents in the six HA teams. It reveals clearly that S4-agents' performances are rather stable, independent of who plays the human partner of S3. We also discovered that the performances of S4s in the H teams do have some relations with human personalities: as time pressure changes, overcautious persons may perform smoothly while impulsive persons may perform inconstantly. However, the last three points in Fig. 6(c) suggests that such difference in personality would not affect the performance much when the time pressure is extremely high.

5. COMPARISON

Other researchers have been investigating (a) computational approaches to RPD to represent human decision making for concept exploration, analysis, or evaluation (e.g., [16]); (b) the use of agents as aids to information gathering in a decision environment (e.g., [7]); and (c) cognitive models of situation awareness (e.g., [2]). The collaborative-RPD model implemented in R-CAST is linked to but also distinguished from the existing work in important ways. First, R-CAST is the first RPD-enabled agent architecture designed for supporting teamwide collaborations (including human-agent and agent-agent collaborations). With collaboration in mind, we take an intensive view of the recognition phase of the RPD process and focus on the investigation of how proactive information exchange among teammates might affect the performance of a decision making team. Second, R-CAST agents can proactively reason across decision spaces, seek missing information from external intelligence sources, exchange relevant information among team-

mates, and monitor an on-going decision against potential expectancy. Third, the "cognitively-aware" agents, as teammates or decision aids, each assigned to a specific functional area, can be used to assist human teams (e.g., military staff) in developing shared situation awareness while balancing information requirements against the dynamic and time sensitive decision making process.

6. CONCLUSION

There has been very little work in incorporating naturalistic decision making models for supporting mixed human-agent teams in making team decisions. In this paper, we described the architecture of R-CAST, the organization of experiences, and how the R-CAST agents make adaptive decisions based on collaborative expectancy monitoring. We evaluated R-CAST in a real-time simulation environment using scenarios with frequent decision-making tasks under different tempo situations. The experiment results conform to psychological findings that human team members are extremely sensitive to their workload in high-tempo situations. More importantly, it suggests that human teams, when supported by R-CAST agents, can perform better in the sense that they can help maintain team performance at acceptable levels in high time pressure situations. While this may not be conclusive yet, it does excite us to further investigate distributed team cognition problems using the R-CAST agents.

Acknowledgments

7. REFERENCES

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

[2] C. Gonzalez, O. Juarez, and J. Graham. Cognitive and computational models as tools to improve situation awareness. In *Proceedings of the 48th Annual Meeting of the Human Factors and Ergonomics Society*, 2004.

[3] E. Hutchins. *Cognition in the wild*. Cambridge, MA: MIT Press, 1995.

[4] G. A. Klein. Recognition-primed decisions. In W. B. Rouse, editor, *Advances in man-machine systems research*, volume 5, pages 47–92. Greenwich, CT: JAI Press, 1989.

[5] G. A. Klein. Recognition-primed decision making. In *Sources of power: How people make decisions*, pages 15–30. MIT Press, 1998.

[6] D. Kleinman, P. Young, and G. Higgins. The DDD-III: A tool for empirical research in adaptive organizations. In *Proceedings of the 1996 Command and Control Research and Technology Symposium*, 1996.

[7] C. Knoblock and J. Ambite. Agents for information gathering. In J. Bradshaw, editor, *Software Agents*, pages 3–29. AAAI/MIT Press, 1997.

[8] T. LaPorte and P. Consolini. Working in practice but not in theory: Theoretical challenges of high reliability organizations. *Journal of public administration research and theory*, 1:19–47, 1991.

[9] J. MacMillan, M. Paley, E. Entin, and E. Entin. Measuring performance in a scaled world: Lessons learned from the distributed dynamic decisionmaking (DDD) synthetic team task. In S. Schiflett, L. Elliott, E. Salas, and M. Coovert, editors, *Scaled Worlds: Development, Validation, and Applications*. Ashgate Publishing Co., 2004.

[10] M. McNeese, E. Salas, and M. Endsley, editors. *New trends in collaborative activities: Understanding system dynamics in complex environments*. Santa Monica, CA: Human Factors and Ergonomics Society, 2001.

[11] E. Norling. Folk psychology for human modelling: Extending the BDI paradigm. In *International Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, pages 202–209, 2004.

[12] E. Norling, L. Sonenberg, and R. Ronnquist. Enhancing multi-agent based simulation with human-like decision making strategies. In S. Moss and P. Davidsson, editors, *Proceedings of the Second International Workshop on Multi-Agent Based Simulation*, pages 214–228, 2000.

[13] E. Salas, T. L. Dickinson, S. A. Converse, and S. I. Tannenbaum. Toward an understanding of team performance and training. In R. W. Swezey and E. Salas, editors, *Teams: Their training and performance*, pages 3–29. Norwood, NJ: Ablex, 1992.

[14] D. Serfaty, E. Entin, and J. Johnston. Team coordination training. In J. Cannon-Bowers and E. Salas, editors, *Making decisions under stress: implications for training and simulation*, pages 221–245. APA Press, 1998.

[15] M. Sierhuis, J. M. Bradshaw, A. Acquisti, R. van Hoof, R. Jeffers, and A. Uszok. Human-agent teamwork and adjustable autonomy in practice. In *7th International Symposium on Artificial Intelligence (I-SAIRAS)*, NARA, JAPAN, 2003.

[16] W. Warwick, S. McIlwaine, R. Hutton, and P. McDermott. Developing computational models of recognition-primed decision making. In *Proceedings of the tenth conference on Computer Generated Forces, Norfolk, VA.*, pages 232–331, 2001.

[17] D. Woods, E. Patterson, and E. Roth. Aiding the intelligence analyst in situations of data overload: A diagnosis of data overload. Technical Report ERGO-CSEL 98-TR-03, Institute for Ergonomics/Cognitive Systems Engineering, The Ohio State University, 1998.

[18] J. Yen, X. Fan, S. Sun, T. Hanratty, and J. Dumer. Agents with shared mental models for enhancing team decision-makings. *Decision Support Systems, Special issue on Intelligence and Security Informatics(in press)*, 2005.

[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.