R-CAST: Integrating Team Intelligence for Human-Centered Teamwork

Xiaocong Fan and **John Yen**
College of Information Sciences and Technology
The Pennsylvania State University
University Park, PA 16802
{xfan, juy1}@psu.edu

Abstract

Developing human-centered agent architectures requires the integral consideration of architectural flexibility, teamwork adaptability, and context reasoning capability. With the integration of various forms of team intelligence including shared teamwork process and progress, dynamic context management and information dependency reasoning, and recognition-primed collaborative decision mechanism, R-CAST offers a flexible solution to developing cognitive aids for the support of human-centered teamwork in information and knowledge intensive domains. In this paper, we present the key features of R-CAST. As evidence of its applications in complex real-world problems, we give two experimental evaluations of R-CAST as teammates and decision aids of human Command and Control teams.

Introduction

Contemporary research on teamwork spans a variety of disciplines, including psychology, cognitive science, artificial intelligence, organization science, concurrent engineering, and business management. Researchers in the field of multi-agent systems have manifested increasing interests in using intelligent agents to model, simulate, and support human teamwork behaviors.

However, developing multi-agent systems for human-centered teamwork is extremely challenging. It mandates the integral consideration of architectural flexibility, teamwork adaptability (to both human and software teammates), and the self-management of collaboration context, which, we believe, are three principles that govern the development of human-centered agent architectures.

First, agent architectures for human-centered teamwork have to conquer domain-specificity by offering flexibility to accommodate teamwork in various domains. The existing architectures have successfully incorporated different forms of architectural flexibility: general model of team synchronization and role-monitoring in STEAM (Tambe 1997), Proxy mechanism in Teamcore (Schurr *et al.* 2006), separation of internal and external representations of shared plans in CAST (Yen *et al.* 2001), to mention only a few. Architectural flexibility is especially important when it is required to form human-agent teams under timing and resource constraints within complex team organizations.

Second, drawn from cognitive studies (Cannon-Bowers, Salas, & Converse 1990), the concept of shared mental models (SMM) has not only been put forward to explain coordinated team behaviors, but also deemed to be one of the key components of any team-oriented agent architecture. Examples are joint intentions (Levesque, Cohen, & Nunes 1990) implemented in STEAM, SharedPlans (Grosz & Kraus 1996) adopted in COLLEGAN (Rich & Sidner 1997), joint responsibilities implemented in GRATE* (Jennings 1995), and Petri nets-based teamwork process realized in CAST. The notion of computational SMM is critical to teamwork adaptability both in coordinating planned team activities and in responding to teamwork failures. However, the development of human-centered collaborative systems, which involve both human-agent and agent-agent collaborations, requires stronger supports for teamwork adaptability. Thus, it is desired to empower an agent architecture with a functional SMM that has a broader scope and a richer structure.

Third, the use of context is of growing importance in developing computational systems that are more responsive to human needs. With a general representation and reasoning capability of teamwork context, an agent can proactively predict others' needs and seek/offer help relevant to the context. However, there are only weak (e.g., teamwork progress in CAST) or no explicit supports for context in the implemented agent architectures. Some of the reasons might be its vague scope (what to cover) and its tight connections with other functional components. We take the position that teamwork context should be at a level higher than SMM; it governs an agent's attention to the very portion of SMM that is relevant to the current activities.

Putting the three principles together would produce successful agent architectures for human-centered teamwork. This paper introduces an attempt toward this vision: R-CAST. R-CAST is a team-oriented cognitive agent architecture built on top of the concept of shared mental models (Cannon-Bowers, Salas, & Converse 1990) in team cognition, the theory of proactive information delivery (Fan, Yen, & Volz 2005), and Klein's Recognition-Primed naturalistic decision model (Klein 1989). Due to the complexity imposed by the three principles, we have chosen to build R-

CAST as cognitive aids for human decision making teams in information and knowledge intensive domains.

R-CAST Architecture

Figure 1 depicts the key components of R-CAST. Below we describe R-CAST from three perspectives: cognitively inspiredness, context awareness, and teamwork readiness.

Cognitively-Inspired Architecture

The RPD Model The original RPD model (Klein 1989) includes a recognition phase and an evaluation phase. In the recognition phase, a decision maker synthesizes the observed features about the current decision situation into appropriate cues or pattern of cues, then uses a strategy called "feature-matching" 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 COA (what actions worked before). In the evaluation phase, a decision maker evaluates the recognized courses of action one by one until a workable solution is obtained. Due to the dynamic and uncertain nature of the environment, a decision maker keeps monitoring the status of expectancy so that the situation can be further diagnosed in case that the decision maker had misinterpreted the situation. Similar to Case-based reasoning, the RPD model stresses on Simon's satisficing criterion (Simon 1955) rather than optimizing in option evaluation.

Iterative RPD Process R-CAST has realized a set of functions corresponding to the main steps of the RPD model: situation investigation, feature matching, recognition evaluation, action implementation, and expectancy monitoring.

Situation investigation is a 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. If the agent is incapable or has insufficient resources to do so, it can send an information request to others. On the other side, other agents who have anticipated the decision maker's information needs may also proactively provide relevant information to the decision maker. While an R-CAST agent gathering information, it 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 experience base and posts 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. *Recognition 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 action may evolve and judge

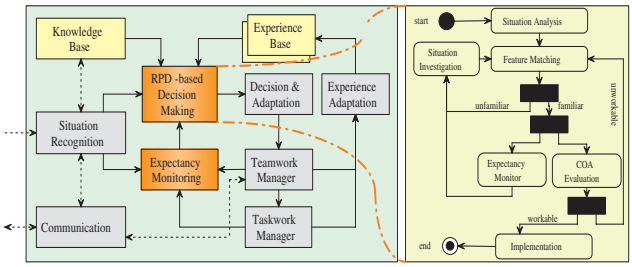


Figure 1: R-CAST Architecture.

whether the relevant goals can be achieved. An R-CAST agent achieves this by checking the constraints (preconditions and effects) associated with the chosen plan (course of action) with respect to its KB. If the preconditions are satisfiable and the effects are consistent, the plan 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.

An *expectancy* states what will happen, serving as a gate-condition for retaining the current recognition. To support adaptive decision making (Serfaty, Entin, & Johnston 1998), an R-CAST agent keeps monitoring the expectancies of a recognition until the completion of the selected course of action. 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.

The RPD model implemented in R-CAST is an *iterative* model, which explicitly incorporates the idea of "recognition refinement", and supports situation reconsideration during action execution phase. R-CAST agents can make a sequence of decisions, with one decision refining the preceding ones, and such a sequence is only restricted by the external time pressure. An agent can always suggest a decision that is acceptable relative to the timing constraints.

Collaborative RPD Process The implemented RPD model in R-CAST is also a "collaborative-RPD" process. An R-CAST agent can not only act as a human's partner, but also team up with other R-CAST agents. Given a specific decision task, each agent of a group can be a decision maker or a supporter, depending whether the agent has the required expertise for making decisions about the task. A decision maker agent (DMA) can derive its information needs regarding the current decision task from the cues considered in the active experience base and the expectancies of those experiences that are found similar to the current situation. The derived information needs can be satisfied in three ways. First, teammates can proactively provide the decision maker agent 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.

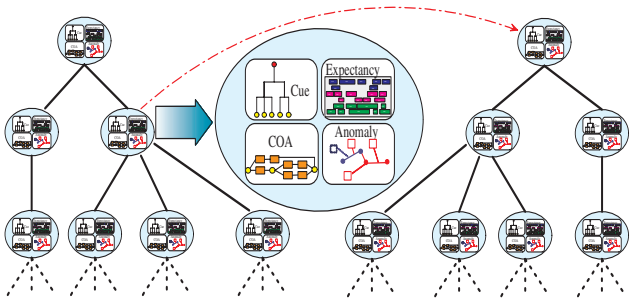


Figure 2: Hierarchical Experience Context.

Context Awareness Architecture

Capturing and using context in decision making is challenging because a desirable context representation should not only support the identification of information relevant to decision making, but also support dynamic changes of context in a way that reflects how a human adapts his/her decisions in a dynamic environment (this may well support human decision makers better than more arbitrary rule-based systems). In doing so, a naturalistic decision making model — Recognition-Primed Decision (RPD) is used to represent the generic *decision-making process context* within R-CAST. To facilitate the reuse of domain knowledge related to decision making, we organize context related to domain knowledge into two separate, but related, representations: *experience context* and *inference context*. Put together, these three types of context representation enable R-CAST to use and integrate various contexts for identifying information relevant to decision making, for adapting decisions to a dynamic environment, and for facilitating reuse of context-related domain knowledge. We next highlight the three important aspects of context representation.

Decision Process Context Multi-context team decision making can be very complex especially when human factors and computing technologies are mutually constraining in their interaction. To reduce the gap between human decision makers and cognitive decision aids (agents), R-CAST is built upon a naturalistic decision making model that captures how domain experts make decisions based on experiences and situational similarity recognition. R-CAST adopts a process context to capture the major phases in the RPD process (including situation recognition, expectancy monitoring, decision adaptation, and COA simulation), and how information is used in these phases.

The decision process context is at a higher level than inference context and experience context; it not only presents its human peer a view of the progress of the current decision making activity, but also governs the selection of the appropriate inference context and experience context. For instance, R-CAST uses the concept of "decision spaces" to organize experiences by decision themes: experiences related to one decision theme are maintained in one decision space or experience knowledge base (EKB). R-CAST can manage multiple decision spaces at run time. The nature of a situation determines what the current working-context is, and situational changes can cause the switch of one context to another. Consequently, an agent working in one decision space can switch into another decision space, and return to a previous space when the current context switched back. This feature allows the implementation of decision aids that can dynamically switch among multiple types of context (e.g., humanitarian, peacemaking, and combat contexts).

Also, the phase of "expectancy monitoring" captures how a human decision maker detects anomalies even after a decision is made so that he/she can re-assess the original decision to determine how to best respond to the anomaly. Anomalous situations can happen frequently in a dynamic environment. For example, suppose under the current situation it is expected that two crowds, G_1 and G_2, should be active in two isolated regions. Anomaly occurs if G_1 and G_2 start to move closer and merge. R-CAST is implemented in a similar way: it can adapt its decision by detecting and responding to situations that are anomalies to the original expectancy (i.e., what is observed/informed conflicts with what was expected).

Hierarchical Experience Context The experience context captures domain-specific experience used by the RPD model, and organizes it in a hierarchical structure (Figure 2) to facilitate the reuse of abstract experience context.

Decision making in human society is typically connected with organizational hierarchies; a higher-level decision-making task usually depends on the results of lower-level decision-making tasks. In order to support multiple-level decision makings (Hollenbeck *et al.* 1995), experiences in an experience base are organized hierarchically in "tree-like" structures: experiences at a higher level node are refined by those at a lower level (with more relevant cues and expectancies considered).

An R-CAST agent uses "recognition anchors" to mark the nodes that it believes are closest in similarity to the current situation. A recognition anchor determines the collection of cues, expectancies, COA, and anomalies that an agent needs to pay attention to. Normally, an R-CAST agent will consider all those cues, expectancies, COA, and anomalies associated with the nodes starting from the root node along the path leading to the recognition anchor being considered. Since an agent may explore multiple ways to gain understanding of a decision situation, it could investigate multiple branches of an experience tree, and thus there may have

multiple active anchors within a decision space. The configuration state of an active decision space can be represented as a bit string marking whether a node is recognitionally anchored by an agent. The "experience context" of an R-CAST agent is composed of the configurations of the active decision spaces, one for each space.

The content of a context is derived from cues, expectancies, COA, and anomalies. Cues are represented as formulas in first-order logic. A cue can be very complex such that the cue itself is the root of an inference tree consisting of many other intermediate or directly-observable types of information. This is typical in real-world domains and it is exactly why collaborative computing and distributed cognition play a key role here, because one single agent/human simply cannot handle the complexity due to lack of sufficient expertise or inference knowledge. Expectancies are represented as formulas in temporal logic. Depending on the importance of a violated expectancy, an agent may choose to adapt the already adopted COA, or backtrack along the experience hierarchy to seek a better recognition. COAs are written in a variant of the process language MALLET (Fan *et al.* 2006a), where each process can be associated with preconditions, statement constraints, and escape conditions. These conditions are first-order formulas; they are evaluated at each time step to determine the collaboration/communication opportunities, whether to pursue a specific teamwork process, and how to respond to teamwork failures. Anomalies are unexpected exceptions; they serve to correct or reject the current situation recognition, and more importantly, they are indicators for switching between different contexts.

Context switching can cause an agent to change its attention from one experience tree to another. The notion of recognition anchors facilitates context freeze and defreeze when an agent needs to switch among multiple active decision tasks in high demanding situations. For example, a Command and Control (C2) team may decide to adjust the military forces allocated to a humanitarian mission to a peacekeeping mission prompting in a nearby location, as more and more spot reports indicate that the situation there might evolve towards undesirable direction. However, the agent has to keep track of the evolvement of both the humanitarian mission and the peacekeeping mission so that it can coordinate activities such as resource allocations and information sharing between the two missions and among other members of the C2 team.

Inference Context The inference context captures inference knowledge that links high-level information needs to lower-level information that can be obtained from information sources or directly observed by agents. An inference context describes the context/situation for drawing various kinds of inference.

The inference context is used in two ways in R-CAST. First, it is used to relate high-level information needs, once they are identified, to low-level information, and identify missing relevant information. Second, the inference context is used to aggregate lower level information, once they are available, to higher level features. Separating the inference context from the process context and the experience context enables a piece of inference knowledge to be used in multiple phases of a process context, and for multiple experiences. For instance, inference knowledge that relates shipping density in an area to geopolitical cost can be used to evaluate a course of action in responding to a threat. The same knowledge can also be used for cue matching (e.g., if the cue of an experience involves geopolitical cost), and for expectancy monitoring.

Teamwork Ready Architecture

R-CAST's teamwork readiness capability is inherited from the CAST architecture (Collaborative Agent for Simulating Teamwork) (Yen *et al.* 2001).

Each agent in a team is initially equipped with a profile of team processes. The processes are then transformed at compile time into Petri-net-like internal structures, which are organized hierarchically so that agents can make inference of collaboration/information needs at an appropriate level. Based on such a computational SMM of team processes, R-CAST offers services for managing dynamic agent/role assignment, for monitoring teamwork progresses, and for initiating context-centric communications.

A teamwork process can be partial in the sense that certain roles (agent variables) need to be determined at run time. This to some extent is similar to the idea of incomplete recipes in the SharedPlans theory (Grosz & Kraus 1996). This issue is tackled in R-CAST by allowing a team to dynamically negotiate on "who works on what" when the team proceeds to a point where the agent-role(task) mapping is still incomplete. Such a point is typically associated with certain constraints for assigning a task(role) to a potential agent. Each team member, based on its own belief base, can propose a set of candidates by stepwisely relaxing the constraints, if possible. It is flexible to use a majority vote or a randomly selected team member to finally establish an agreement on the agent assignment.

Inter-agent communication and its management is also a crucial part of the R-CAST architecture. An agent is required to inform other teammates of its task progress at critical points. For instance, a group of agents involved in a team process have to send synchronization messages to make sure everyone is ready before executing the process. By allowing agents to exchange information regarding their teamwork progress, they can have a global picture of the dynamic progress of the committed team activity. This enables agents to determine how their individual actions fit together, to act proactively to achieve coherent teamwork, and to anticipate the opportunities of offering help. Knowing of teamwork progress is also very useful in developing shared situation awareness when the communications bandwidth is limited. Teamwork progress can be exploited such that an agent can progressively update the status of other team members' information needs. An agent can thus offer timely help without disturbing the other agents with information no longer relevant to their activities.

Integration of Team Intelligence

The integrative aspects of R-CAST is two-fold. First, as described in Section 1, developing human-centered agent ar-

chitectures requires the integral consideration of architectural flexibility, teamwork adaptability, and context reasoning capability. However, actually implementing systems conforming to these high-level principles is a significant challenge, which R-CAST meets.

R-CAST employs a knowledge-based approach to achieve architectural flexibility. All R-CAST components are integrated around its internal knowledge base (KB). Specifically, KB is used in cue-matching to identify missing information, in expectancy monitoring to check the validation of situation recognition, in COA evaluation to check the feasibility of a course of action, and in team process execution to check the readiness of all team members involved. Team plans are described in a knowledge representation language called MALLET; this facilitates reuse of generic process knowledge across domains. R-CAST components can be configured using a profile, which allows the replacement of certain components to serve different studies.

R-CAST offers two approaches to achieve teamwork adaptability. First, it supports a functional SMM with a richer structure that not only covers team structures and team processes used by agents to infer collaboration needs, but also covers dynamic teamwork contexts. Such an enhanced representation of shared mental models enables an R-CAST agent to better manage its own 'focal' attention and to initiate human-agent collaboration in a human-appreciatable way. Second, non-trivial collaborative multi-agent systems need to continuously make decisions, which demands a group of agents to coordinate not only on domain-specific tasks but also in the decision-making process itself. Meshing humans' decision making process with agents' decision making process promises better human-agent collaboration. However, it requires humans and agent teammates to maintain a shared understanding of the decision making progress. To this end, R-CAST has incorporated the recognition-primed decision model (RPD)— a naturalistic decision making process that may well support human decision makers better than more arbitrary rule-based systems.

R-CAST supports context reasoning by distinguishing decision process context, experience context and inference context. These three types of context representation together enable R-CAST agents to reason about contexts for reusing inference knowledge and experience across multiple contexts, for identifying information relevant to decision making, and for adapting decisions to a dynamic environment.

Second, R-CAST is a system reflecting the synergy of cognitively inspiredness, context awareness, and teamwork readiness. Its cognitively-inspiredness lies in the using of Klein's naturalistic decision making model as the glue of teamwork; its context awareness is supported by three forms of context representation: decision process context, experience context, and inference context; and its teamwork readiness manifests in the Petri Nets-based representation of teamwork processes and capabilities of monitoring progress. These three aspects are tightly integrated. For instance, the RPD process breaks decision making into two phases, recognition and evaluation, each of which maps to a set of functions in R-CAST, for situation investigation, feature matching, recognition evaluation, implementation, and expectancy

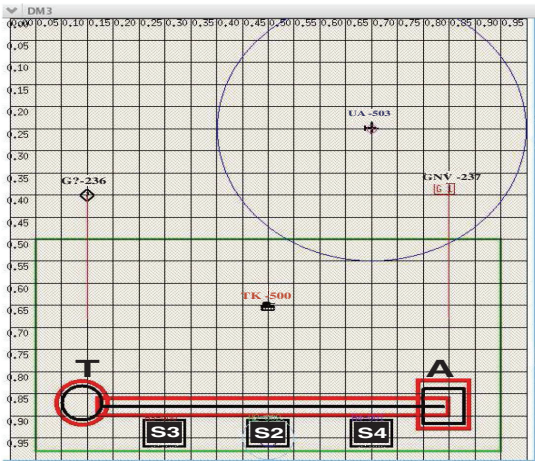


Figure 3: Use R-CAST to adapt decisions.

monitoring. These functions trigger context management, as well as drive information processing and proactive information sharing, to orchestrate team activities including both decision making and domain tasks.

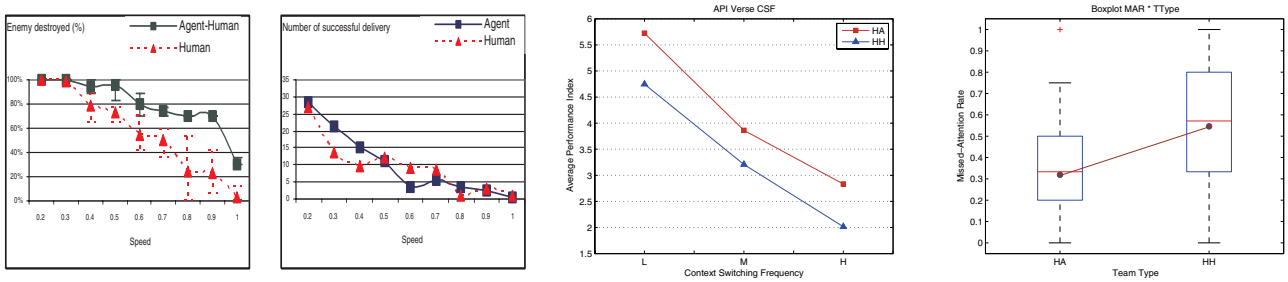
Applications

The integration of team intelligence within R-CAST offers at least two benefits. First, R-CAST provides a flexible solution to developing decision aids for supporting human decision making teams under time stress. Second, R-CAST can act as virtual teammates of human team members in information and knowledge intensive domains. The benefits can be buttressed by its applications as described below.

The capability of R-CAST as decision aids was evaluated in a simulated battlefield (Fan *et al.* 2005). We used the DDD (Distributed Dynamic Decisionmaking) environment (Kleinman, Young, & Higgins 1996) as the testbed, and designed scenarios that involve a blue (friendly) force consisting of three battle functional areas (BFAs): the intelligence cell (S2), the operations cell (S3), and the logistics cell (S4).

As shown in Figure 3, in the frontier of a battlefield, there is a supply route connecting an airport A and a target area T. The roles of the BFAs were simplified as follows: S2 controls a UAV to collect information and identify whether an approaching object (task) is a neutral force or enemy unit; S3 controls a tank to destroy enemies and protect the supply route; and S4 controls a truck to deliver supplies from A to T (the truck will be destroyed if it is within the attack range of an approaching enemy). 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. The BFAs have to collaborate with each other in order to have a good performance.

An experiment was conducted to understand whether R-CAST could enhance the performance of Command and Control (C2) teams under time stress. Two types of teams were used: H teams consisting of 3 human subjects playing the roles of S2, S3 and S4; HA teams consisting of 3 R-CAST agents playing the roles of S2, S3 and S4, with the S3 agent paired with a human subject. HA Teams involved



(a) (b) (c) (d)

Figure 4: (a) Enemies destroyed by S3; (b) Supplies delivered by S4. (c) API verse CSF; (d) Boxplot of MAR by TType.

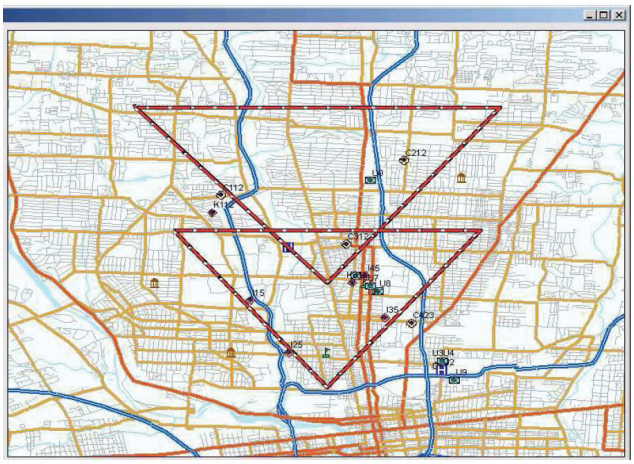


Figure 5: Use R-CAST to address 3-block challenge.

both human-agent and agent-agent collaborations.

Fig. 4(a) and 4(b) plot the average performance in terms of percentage of enemy destroyed and times of successful delivery of supplies. While Fig. 4(b) does not show much difference between H teams and HA teams regarding successful delivery, Fig. 4(a) clearly indicates that human-agent teams performed much better than pure human teams regarding enemy destroyed: 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 also increased. The result confirmed that R-CAST agents as decision aids could enhance the performance of C2 teams under varying time pressure in handling threats.

Another study (Fan *et al.* 2006b) was conducted to evaluate whether the context-awareness feature of R-CAST could help C2 teams to make real-time decisions involving multiple dynamic contexts. Still involving S2, S3 and S4 roles, C2 teams were assumed to operate in complex urban terrain, reacting to potential threats associated with crowds, IEDs (Improvised Explosive Device), and key insurgents. It was named as '3-block challenge' because within three-block area in a city officers in command must react to a constant flow of intelligence reports and make timely decisions for three kinds of tasks including combat, peacekeeping and humanitarian missions. Figure 5 is a screen-shot showing a map display that S2 and S3 human subjects used during the experiments to facilitate the assessment of threat locations and the allocation of friendly units to handle threats.

A $2 \times 3 \times 3$ factorial treatment design was used, involving three factors: team type, context switching frequency (CSF), and task complexity. Two types of C2 teams were formed from 30 subjects randomly recruited from a US Army ROTC (Reserve Officer Training Corps) organization: S2H-S3H teams (each has a subject playing S2 and a subject playing S3), and S2A-S3HA teams (each has an R-CAST agent playing S2 and a subject together with an R-CAST agent playing S3). The context switching frequency and task complexity each has three levels, varying the frequency of updating a C2 team with field information and the number of active threats in the field, respectively.

Fig. 4(c) plots the API (Average Performance Index) performance of HA and HH teams under difference levels of CSF. It shows a performance drop for both HH and HA teams as the context switching frequency increased: both suffered more under more time-stressed situations. However, HA teams performed significantly better than HH teams at each CSF level. The performance of a secondary task (a subject's attention paid to the key buildings nearby the target being tasked) is plotted in Fig. 4(d), which shows that HA teams missed significant less attentions than HH teams. Overall, this experiment demonstrated that R-CAST agents, as both teammates (S2) and decision aids (S3), can play a critical role in alleviating the impact of human's cognitive capacity on the performance of decision making tasks involving multiple contexts.

Related Work and Summary

Among related work, STEAM (Tambe 1997) and COLLAGEN (Rich & Sidner 1997) are two collaborative agent architectures built on the joint intentions theory (Levesque, Cohen, & Nunes 1990) and the SharedPlans theory (Grosz & Kraus 1996), respectively.

STEAM is built on top of the Soar architecture (Laird, Newell, & Rosenbloom 1987), using joint intentions as a building block to hierarchically build up the mental attitude of individual team members. R-CAST differs from STEAM in several important aspects. First, while teamwork in STEAM is realized by building up a hierarchy of joint intentions, R-CAST (CAST) employs a shared mental model of team processes to coordinate team activities and infer collaboration needs. Its agent-binding capability borrows from the idea of 'plan evolution' of the SharedPlans formal-

ism. Second, teamwork in R-CAST not only manifests in its mechanism of shared team processes, it also reflects at a meta-level in the sense that team members can collaborate along the recognition-primed decision process.

COLLAGEN, built on top of the principles that underlie human collaboration in discourse theory, realizes a standard mechanism for maintaining the flow and coherence of interactions between a human user and an intelligent agent. While both R-CAST and COLLAGEN have a root in the SharedPlans theory, R-CAST focuses more on the role of cognitive aids to human decision making teams.

The proxy-based DCI architecture (Schreckenghost *et al.* 2002) has been proposed for supporting multiple humans interacting with multiple automated control agents. However, it is unclear how DCI allows agents to respond to novel situations and manage multiple contexts. Such functionalities are gracefully offered by R-CAST due to the synergistic incorporation of the RPD model. Another related work is the DEFACTO system (Schurr *et al.* 2006), which provides a multiagent environment for training incident commanders. While R-CAST can also be used for training purpose, the vision is to act as cognitive aids in the real fields.

In sum, with the integration of various forms of team intelligence including shared teamwork process and progress, dynamic context management and information dependency reasoning, and recognition-primed collaborative decision mechanism, R-CAST offers a flexible solution to developing cognitive aids for supporting human-centered teamwork in information and knowledge intensive domains. Intelligence analysts need tools and techniques to help protect themselves from avoidable errors (Heuer 1999). Our studies and the experiments conducted so far demonstrated that R-CAST agents can serve as one such tool to achieve reduced cognitive load, enhanced situation awareness, and positive human-agent collaboration.

Acknowledgments

This work is supported as an FY06 Research Task under the Army Research Laboratory's Advanced Decision Architectures Collaborative Technology Alliance (ARL ADA CTA).

References

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

Fan, X.; Sun, S.; McNeese, M.; and Yen, J. 2005. Extending the recognition-primed decision model to support human-agent collaboration. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, 945–952. ACM Press.

Fan, X.; Yen, J.; Miller, M.; Ioerger, T.; and Volz, R. A. 2006a. MALLET–a multi-agent logic language for encoding teamwork. *IEEE Transaction on Knowledge and Data Engineering* 18(1):123–138.

Fan, X.; Sun, B.; Sun, S.; McNeese, M.; and Yen, J. 2006b. RPD-enabled agents teaming with humans for multi-context decision making. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 34–41. ACM Press.

Fan, X.; Yen, J.; and Volz, R. A. 2005. A theoretical framework on proactive information exchange in agent teamwork. *Artificial Intelligence* 169:23–97.

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

Heuer, R. J. 1999. *Psychology of Intelligence Analysis*. Center for the Study of Intelligence.

Hollenbeck, J.; Ilgen, D.; Sego, D.; Hedlund, J.; Major, D.; and Phillips, J. 1995. The multi-level theory of team decision-making: Decision performance in teams incorporating distributed expertise. *Journal of Applied Psychology* 80:292–316.

Jennings, N. R. 1995. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence* 75(2):195–240.

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

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

Laird, J.; Newell, A.; and Rosenbloom, P. 1987. Soar: an architecture for general intelligence. *Artificial Intelligence* 33(1):1–64.

Levesque, H. J.; Cohen, P. R.; and Nunes, J. 1990. On acting together. In *Proceedings of AAAI-90*, 94–99.

Rich, C., and Sidner, C. 1997. Collagen: When agents collaborate with people. In *Proceedings of the International Conference on Autonomous Agents (Agents'97)*, 284–291.

Schreckenghost, D.; Martin, C.; Bonasso, P.; Kortenkamp, D.; Milam, T.; and Thronesbery, C. 2002. Supporting group interaction among humans and autonomous agents. *Connection Science* 14(4):361–369.

Schurr, N.; Patil, P.; Pighin, F.; and Tambe, M. 2006. Using multiagent teams to improve the training of incident commanders. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 1490–1497.

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

Simon, H. 1955. A behavioral model of rational choice. *Quarterly Journal of Economics* 69:99–118.

Tambe, M. 1997. Agent architectures for flexible, practical teamwork. In *National Conference on Artificial Intelligence (AAAI)*, 22–28.

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