MokSAF: How should we support teamwork in human-agent teams?

Terri L. Lenox†, Terry Payne, Susan Hahn†,
Michael Lewis† & Katia Sycara

CMU-RI-TR-99-31

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

† University of Pittsburgh
School of Information Sciences
Dept. of Information Sciences & Telecommunications
135 N. Bellefield Ave.
Pittsburgh, PA 15260

September 1999

This research has been supported by the Office of Naval Research, ONR grant N-00014-96-1-1222. Views and conclusions contained in the document are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of the Office of Naval Research or the United States Government.

KEYWORDS

Teams, teamwork, agents, interfaces and planning.

ABSTRACT

In this paper, we describe an interface agent, two different route planning agents and a pilot study which examined whether these agents could support a team planning task. The MokSAF *interface agent links an Artificial Intelligence (AI) route-planning agent to a Geographic Information System (GIS). The user specifies a start and an end point and the route-planning agent finds a minimum cost path between the points. The user is allowed to define additional "intangible" constraints (not due to terrain characteristics) corresponding to geographic regions, which can be used to steer the agent's behavior in a desired direction. A second agent (the naive route planning agent, or* Naive RPA*) has access to the same knowledge of the terrain and cost functions available to the* Autonomous RPA*, but uses this knowledge to critique paths specified by the user. We hypothesize that as the complexity of intangible aspects of a planning problem increase, the* Naive RPA *will improve in relative performance. The reported study found advantages across the board for the* Autonomous RPA *in a team-planning task.*

INTRODUCTION

As the task environment becomes more complex and uncertain and the time frame for making decisions is shortened, reliance on computer-assisted decision-making by both individuals and teams has increased dramatically. The current trend is towards software that not only retrieves information upon request but also intelligently anticipates, adapts and actively seeks ways to support users [1]. These software agents can reduce the amount of interaction between humans and the computer system and allow the humans to concentrate on other activities such as assessing the situation, making decisions, or reacting to changes in the system [2].

These gains, however, come at the cost of increasing complexity and/or confusion in our relation with software. The management skills of decomposing and delegating tasks and monitoring performance once reserved for human subordinates may become necessary for interacting with sophisticated agents. Conversely, those agents which shield us from complex interactions by quietly looking over our shoulders to anticipate our actions may actually decrease our situational awareness leaving us uncertain as to what is being done on our behalf [3]. These difficulties can be compounded where multiple agents and humans are required to work as a team. Under these conditions, cascading delegation among software agents and unknown silent assistance complicates the already challenging task of cooperating, communicating, and monitoring the task and other team members.

Our research focuses on active (agent critiquing) and passive (agent performance) techniques, which enable us to communicate with software agents. While much of the early focus on decision aids has been on supporting the individual [4], we examine the middle ground of individually controlled software agents used in team tasks.

Although it is desirable to organize individuals into groups and provide support via software agents, this is not necessarily an easy task. Multiple software agents, working in teams, can autonomously sort through and evaluate the enormous quantities of information available to a team and thus, free it for other crucial tasks. Incorporating software agents into human teams presents many challenges. What roles should agents play in the overall team context? Can these roles be adapted during task performance? What are effective ways for software agents to interact with the human team members and with each other so as to increase team effectiveness? What are the appropriate measures of agent effectiveness within a team context and of team effectiveness?

TEAMS AND TEAMWORK

Characteristics of successful teams include self-awareness, within-team interdependence, feedback, performance monitoring, clear communication of intentions, and assisting other team members when necessary. A team can be defined as [9]:

> "... a distinguishable set of two or more people who interact dynamically, interdependently, and adaptively towards a common and valued goal/objective/ mission, who each have been assigned specific roles or functions to perform, and who have a limited life-span of membership."

Team members must have a shared understanding of the capabilities, goals and intentions of other members in order to function effectively [10]. This shared understanding helps teammates to predict each other's performance under normal and specific circumstances. Typically, they gain this understanding through experience and training with the system [11].

To contribute to team success, software agents must support these forms of group interaction as well as more task-oriented functions. The potential impact of successful development and deployment of agent technologies to mission critical teams includes:

1) Reducing the time to make a decision;

2) Allowing teams to consider a broader range of alternatives;

3) Allowing teams to manage contingencies flexibly by rapidly re-planning;

4) Reducing the time required for a team to form a shared mental model of the situation;

5) Reducing both individual and team errors;

6) Increasing the cohesion among team members;

7) Increasing overall team performance.

To be successful, team members must understand how to interact and control the computer technologies. They must know how to gather, summarize and interpret the information necessary to perform the task(s). In addition, team members must understand their role in the task and what information is required by their teammates. Finally, they should be aware of and act in accordance with the strengths and weaknesses of their teammates [5]. We believe that properly designed software agents can alleviate some of the burden from the human members of the team.

USING THE INFOSPHERE TO MAKE PLANS

Human decision-makers, particularly military commanders, typically face time pressures and an environment where changes may occur in the task, division of labor, and allocation of resources. Information such as terrain characteristics, location and capabilities of enemy forces, direct objectives and doctrinal constraints are part of the commander's *infosphere*. Information within the infosphere has the opportunity for data fusion, situation visualization, and "what-if" simulations. Software agents have access to all information in the infosphere and can plan, criticize, and predict the consequences of actions using the infosphere information at a greater accuracy and finer granularity than the human commanders can. Multiple agents can be designed to use information cooperatively in the infosphere to satisfy specified goals.

However, these agents cannot consider information outside the infosphere unless it is captured in physical terms. This extra-infosphere data consists of intangible or multiple objectives involving morale, the political impact of actions (or inaction), intangible constraints, and the symbolic importance of different actions or objectives. Military commanders, like other decision-makers, have vast experiential information that is not easily quantifiable. Commanders must deal with idiosyncratic and situation-specific factors such as non-quantified information, complex or vaguely specified mission objectives and dynamically changing situations (e.g., incomplete/changing/new information, obstacles, and enemy actions). When participating in a planning task, commanders must translate these intangible constraints into physical ones to interact with planning agents.

The issue then becomes how software agents should interact with their human team members to incorporate these intangible constraints into the physical environment effectively.

TEAM APPROACHES

As the role of teams becomes more important in organizations, developing and maintaining high performance teams has been the goal of several researchers [12,13]. One major question is how to turn a team of experts into an expert team. There are several strategies emerging, including task-related cross training [13] and integrating software agents into human-agent teams, which is the focus of this research. We have developed a framework for examining the different ways that software agents can be deployed in support of team performance:

- Support the individual team members in completion of their own tasks;

- Allocate an agent its own subtask as if we were introducing another member into the team;
- Support the team as a whole.

The first option focuses on the specific tasks that an individual must accomplish as part of the team. For a second option, all the issues associated with communication and coordination among team members become relevant [4,6,7]. The third option involves facilitating communication, allocating tasks, coordinating the human agents, and focusing attention. Specifically the focus is on how software agents can be used to support and promote teamwork. There have been several team models developed by researchers; the one selected for this research is best described by Cannon-Bowers and Salas [5] and Smith-Jentsch, Johnston and Payne [7].

This teamwork model consists of four dimensions that build and maintain situational awareness within the team and hence support effective performance:

- Information exchange - exploit all available information sources; disseminate information; provide situation updates;
- Supporting behavior - prompt correction of team errors; provide and request backup when necessary;
- Communication - proper terminology; complete internal and external reports; brevity and clarity;
- Team initiative/leadership - provide feedback to team members; state clear and appropriate priorities.

This model focuses on observable, measurable behavior that can be evaluated and used to train teams to be more effective. A basic tenet of this model is that teamwork skills are different from task-based competencies. The performance of teams, especially in tightly coupled tasks, is believed to be highly dependent on these interpersonal skills.

In previous studies, we used a low fidelity radar simulation environment called Tandem [8] to examine whether agents should support an individual's performance or the team's performance in a target identification task. Three-person teams were provided with one of three different aiding conditions. The first agent, the *Individual Agent*, aided the individual task and assisted communication among team members by aggregating values. This agent showed all data items available to an individual team member and filled in the values for the data items as the participants selected them from a menu. The second agent, the *Team Clipboard Agent*, aggregated values from all members and automatically passed values as they were selected from the menu to the appropriate team member. The third agent, *Team Checklist*, aided team coordination by displaying who had access to what data. Teams were asked to identify a series of targets on the radar screen. These targets varied in how difficult they were to identify. That is, easy targets had no ambiguity on five pieces of identification data; medium targets had ambiguity on one or two data items out of five possible data items; and hard targets were ambiguous on two out of five items. We found that aiding teams helped more than aiding individuals when the team was faced with hard targets.

THE PLANNING ENVIRONMENT: *MOKSAF*

A computer-based simulation called *MokSAF* has been developed to evaluate how humans can interact and obtain assistance from agents within a team environment. *MokSAF* is a simplified version of a virtual battlefield simulation called ModSAF (Modular Semi-Automated Forces). *MokSAF* allows two or more commanders to interact with one another to plan routes in a particular terrain. Each commander is tasked with planning a route from a start point to a shared rendezvous point by a certain time. The individual commanders must then evaluate their plans from a team perspective and iteratively modify these plans until an acceptable team solution is developed.

The interface agent that is used within the *MokSAF* Environment is illustrated in Figure 1. This agent presents a terrain map, a toolbar, and details of the team plan. The terrains displayed on the map include soil (plain areas), roads (solid lines), freeways (thicker lines), buildings (black dots), rivers and forests. The rendezvous point is represented as a red circle and the start point as a yellow circle on the terrain map. As participants create routes with the help of a *route-planning agent* (see below), the routes are shown in bright green. The second route shown is from another *MokSAF* commander who has agreed to share a route. The partially transparent rectangles represent intangible constraints that the user has drawn on the terrain map. These indicate which areas should be avoided when determining a route.

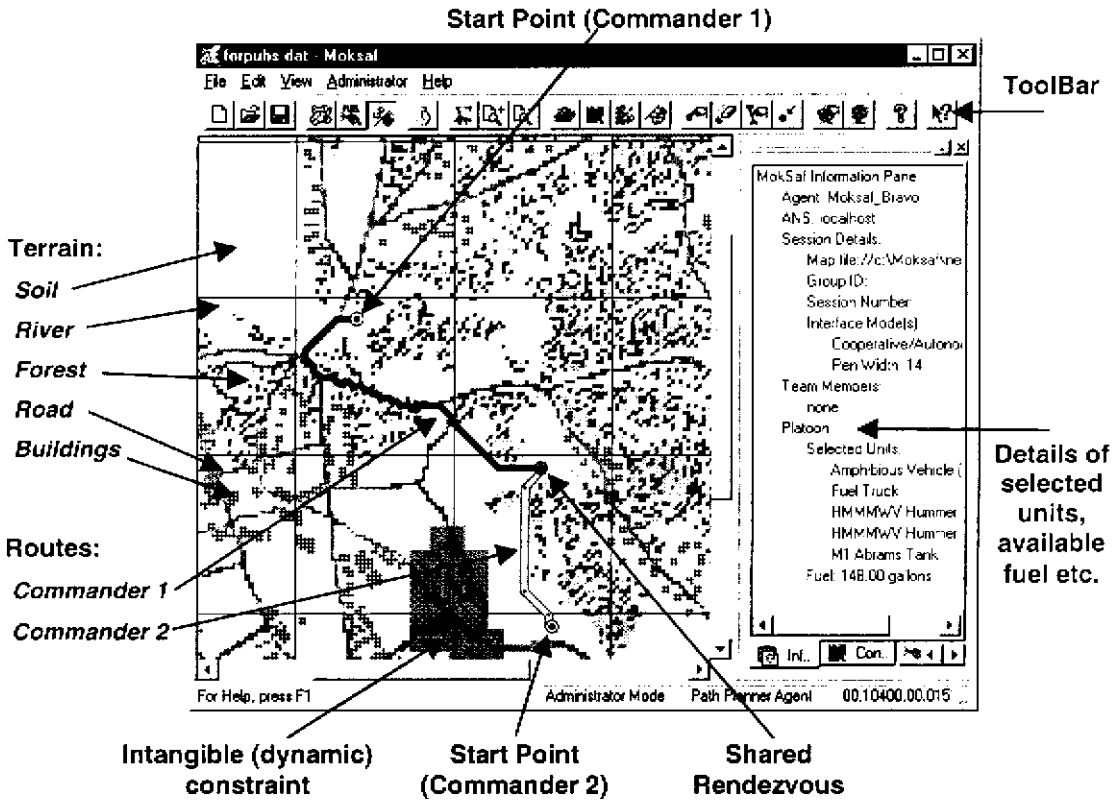

Figure 1: The MokSAF Interface Agent

ROUTE-PLANNING AGENTS

Two different *route-planning agents* (*RPAs*) have been developed which interact with the human team members in the planning task. The first agent, the *Autonomous RPA*, guides the human team members through the route-planning task and performs much of the task itself. This agent acts much like a "black box". The agent creates the route using its knowledge of the physical terrain and an artificial intelligence planning algorithm that seeks to find the shortest path. The agent is only aware of physical constraints, which are defined by the terrain map and the platoon composition, and intangible constraints, which are specified by the commanders.

The second agent, the *Naive RPA*, analyzes the routes drawn by the human team members and helps them to refine their plans. In this mode, the human and agent work jointly to solve the problem (e.g. plan a route to a rendezvous point). The system was designed so that the workload is shared between the different components (agent or human) according to each component's relative strengths. Thus, the

commander, who has a privileged understanding of the intangible constraints and utilities associated with the mission, can direct the route around these constraints as desired. However, the commander may not have detailed knowledge about the terrain, and so the agent can indicate where the path is sub-optimal due to violations of physical constraints.

The commander draws the desired route and requests that the *Naive RPA* review the route for physical violations or to indicate ways in which the path could be improved. The commander can iteratively improve the plan until a satisfactory solution is reached.

EXPERIMENTAL METHODOLOGY

In the *MokSAF* pilot experiments, a deliberative, iterative and flexible planning task is examined. There are three commanders (Alpha, Bravo and Charlie), each with a different starting point and a common rendezvous point. Each commander selects units for his/her platoon from a list of available units. This list currently contains M60A3 tanks, M109A2 artillery units, M1 Abrams tanks, AAV-7 amphibious assault vehicles, HMMWVs (i.e., hummers), ambulances, combat engineer units, fuel trucks and dismounted infantry. It can be easily modified to add or delete unit types. With the help of one of the *RPAs*, each commander plans a route from a starting point to the rendezvous point for the specified platoon.

Once a commander is satisfied with the individual plan, he/she can share it with the other commanders. Teammates needed to communicate with one another to complete their tasks successfully. Conflicts can arise due to several issues including shared routes and/or resources and the inability of a commander to reach the rendezvous point at the specified time. The mission supplied to the commanders provides them with a final total of vehicles required at the rendezvous point. They must coordinate regarding the number and types of vehicles they are planning to take to the rendezvous point. In addition, the commanders are told that they should not plan a route that takes them on the same path as any other commander and that they should coordinate their routes to avoid shared paths.

Materials

MokSAF 2.0 was used for this pilot study. It consists of an interface agent that presents the commander with a standard terrain map and markings, a toolbar as seen in Figure 1, a communication window where commanders can send and receive messages and share plans, and a constraint tree. The two different *route-planning agents* described above were evaluated.

Participants

Fifteen teams consisting of three-persons were recruited (10 teams who used the *Autonomous RPA*, and five who used the *Naive RPA*) from the University of Pittsburgh and Carnegie Mellon University communities. Participants were recruited as intact teams, consisting of friends or acquaintances.

Procedures

Each team participated in a 90-minute session that began with a 30-minute training session in which the *MokSAF* environment and team mission were explained. The team was told to find the optimal path between the start and rendezvous points, to avoid certain areas or go by other areas, to meet the mission objectives for numbers and types of units in their platoon, and to avoid crossing paths with the other commanders. After the training session, the team participated in two 15-minute trials. Each trial used the same terrain, but different start and rendezvous points and different platoon requirements. At the conclusion, participants were asked to complete a brief questionnaire.

We are measuring individual and team performance with respect to the planning task, and using a cognitive work analysis technique to analyze the interaction among the team members to determine if and how each type of agent supports the team as a whole. One question we hope to answer is which

interface type best supports the overall team performance in this type of task. There are two expected trade-offs between the *Autonomous RPA* (which acts as an oracle) and the *Naive RPA* (which acts as a critic):

1) The complexity of intangible constraints and multiplicity of goals;
2) The time and/or quality of the agent-generated solutions (*Autonomous RPA*) versus the agent-critiqued solutions (*Naive RPA*).

RESULTS

We examined time to share a route for the three commanders and found that the *Autonomous RPA* had an advantage over the *Naive RPA* ($p < .005$ for Alpha, $p < .063$ for Bravo and $p < .006$ for Charlie). Groups using the *Autonomous RPA* spent less time creating their individual plans before sharing them with their teammates (Tables 1 and 2) These results are illustrated in Figure 2.

Table 1: Time to Shares Route in Trial 1

Route-Planning Agent	*Alpha Trial 1*	*Bravo Trial 1*	*Charlie Trial 1*
Autonomous	5.17	5.2	5.4
Naive	8.69	9.57	6.7

Table 2 : Time to Share Routes in Trial 2

Route-Planning Agent	*Alpha Trial 2*	*Bravo Trial 2*	*Charlie Trial 2*
Autonomous	2.9	4.01	4.4
Naive	7.1	6.15	8.7

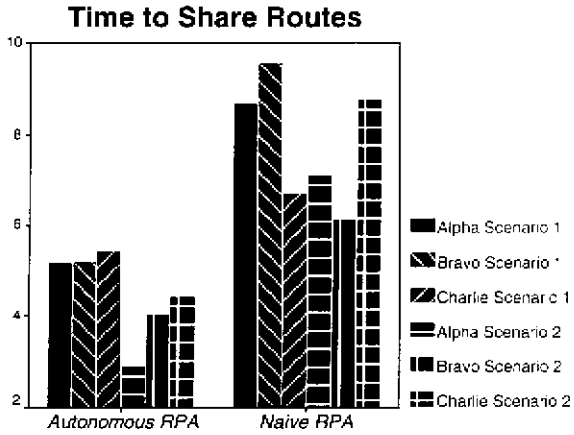

Figure 2 Time to Share Routes

We also examined the individual path lengths for each commander at two points in each trial - when routes were first shared with the team and at the end of the 15-minute trial (Tables 3 and 4). The ending path lengths for Alpha ($p < 0.000$), Charlie ($p < .000$) and combined ($p < 0.000$) were better using the *Autonomous RPA* than with the *Naive RPA* (see Figure 3).

Table 3: Ending Path Length in Trial 1

Route-Planning Agent	*Alpha Trial 1*	*Charlie Trial 1*	*Total Trial 1*
Autonomous	79.7	7.8	181.6
Naïve	153.8	37.8	282.2

Table 4: Ending Path Length in Trial 2

Route-Planning Agent	*Alpha Trial 2*	*Charlie Trial 2*	*Total Trial 2*
Autonomous	31.1	53.5	114.8
Naïve	77.4	91.6	210.6

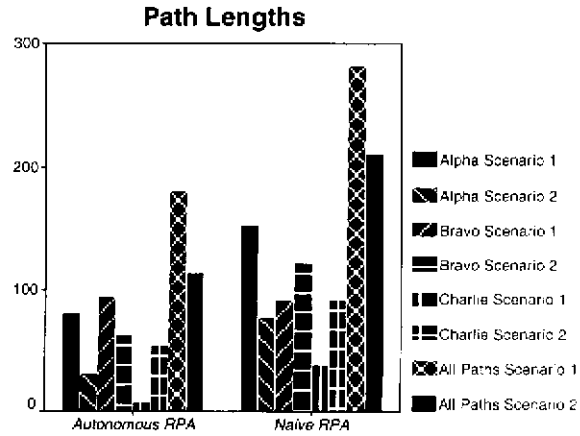

Figure 3: Ending Path Lengths

It is expected that path lengths between the first time a route was shared and at the end of a trial would vary due to issues related to conflict resolutions among the teammates. There was a significant difference in the change in path lengths from these two points in time ($p < .018$). Table 5 (and Figure 4) shows that participants using the *Naive RPA* made more changes in their paths. This change could be due to the state of the route when it was first shared; that is, the routes drawn by the participants may have required additional refinement during the trial. Another possible reason for the change in the paths could be due to interactions with teammates.

Table 5: Change in Path Lengths from First Share to End of Trial

Route-Planning Agent	*Trial 1*	*Trial 2*
Autonomous	130.4	37.1
Naïve	222.2	82.8

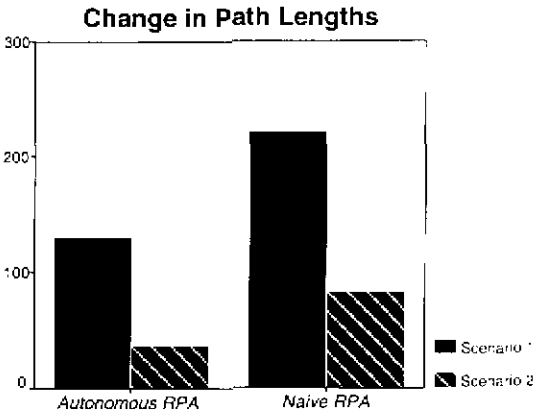

Figure 4: Change in Path Lengths

Participants were asked to create optimal routes given certain confounding factors (e.g., avoiding constraints, going to designated areas, and avoiding traveling on the same paths as other commanders). They were also asked to plan as a group numbers and types of units at the rendezvous point. We found that there was no difference in this selection of units in either *route-planning agent*.

DISCUSSION

In its current form, the *Autonomous RPA* has been shown to provide better assistance for both individual route planning and team-based re-planning. While the individual plans for *Naive RPA* users in the Alpha and Bravo roles were not significantly different from *Autonomous RPA* users in quality, it took them substantially more time to construct their routes. The eventual coordinated routes were uniformly better for each of the individual positions in the *Autonomous RPA* group and for the team as a whole.

Despite this clear superiority, participants in the *Autonomous RPA* group frequently expressed frustration with the indirection required to arrange constraints in the ways needed to steer the agent's behavior and often remarked that they wished they could "just draw the route by hand".

Comments on the *Naive RPA* focused more closely on the minutiae of interaction. In its current form, the user "draws" a route on the *interface agent* by specifying a sequence of points at the resolution of the terrain database. To do this, the user clicks to specify an initial or intermediate point in the path and then clicks again at a second point. A sequence of points is then drawn in a straight line between these locations. A route is built up incrementally by piecing together a long sequence of such segments. Although tools are provided for deleting unwanted points and moving control points, the process of manually constructing a long route is both tedious and error prone. While interaction with the *Autonomous RPA* automatically avoids local obstacles such as trees and closely follows curves in roads due to their less costly terrain weights, a user constructing a manual route is constantly fighting unseen obstacles which void her path or line segments which stray a point or two off a road into high penalty terrain. The anticipated advantages of heuristic planning and cooperation among human users were largely lost due to the necessity of focusing on local rather than global features of routes. Rather than zooming in and out on the map to see the start and rendezvous points before beginning to draw, our subjects were forced to work from the first at the highest magnification in order to draw locally correct segments. The resulting problems of maintaining appropriate directions across scrolling segments of a

map are not dissimilar to hiking with a compass. Although you can generally move in approximately the right direction you are unable to take advantage of features of the terrain you might exploit if a more global view were available.

Of the lessons learned in this initial test of our agent-based alternatives, the difficulty of creating good interfaces for communicating human intent stands out. The *Autonomous RPA*, which minimizes the human-communication, was very successful in its initial implementation. The *Naive RPA*, by contrast, will require substantial revision before it approaches the planner in articulatory directness and fluency. We hope that subsequent refinements to the *Naive RPA* may allow a more thorough comparison of the effects of agent and human initiative on team planning and re-planning tasks.

ACKNOWLEDGEMENTS

This research was supported by an Office of Naval Research grant N-00014-96-1-1222. Special thanks go to Constantine Domashnev, Glenn Lautenbacher and Martin van Velsen for their assistance in the development of this software.

REFERENCES

1. Bradshaw, J. M. (1997). "Introduction," in J. M. Bradshaw (ed.) *Software Agents*. AAAI Press: Menlo Park, CA, 3-48.

2. Zachary, W., Le Mentec, J-C., and Ryder, J. (1996). "Interface agents in complex systems," In C. A. Ntuen and E. H. Parks (Eds), *Human Interaction with Complex Systems: Conceptual Principles and Design Practice*, Kluwer Academic Publishers.

3. Lewis, M. (1998). "Designing for human-agent interaction," *AI Magazine*, Summer 1998, 67-78.

4. Roth, E. M., Malin, J. T. and Schreckenghost, D. L. (1997). "Paradigms for Intelligent Interface Design." In M. Helander, T. K. Landauer, and P. Prabhu (Eds), *Handbook of Human-Computer Interaction*, Second Edition. 1177 - 1201.

5. Cannon-Bowers, J.A., and Salas, E. (1998). "Individual and Team Decision Making Under Stress: Theoretical Underpinnings." In J.A. Cannon-Bowers, and E. Salas (Eds.) *Making Decisions Under Stress: Implications for Individual and Team Training*. American Psychological Association: Washington, D.C.

6. Jones, P. M. and Mitchell, C. M. (1995). "Human-computer cooperative problem solving: Theory, design, and evaluation of an intelligent associate system." *IEEE Transactions on Systems, Man, and Cybernetics*. SMC-25(7), 1039-1053.

7. Smith-Jentsch, K. Johnston, J. H., and Payne, S. C. (1998). "Measuring team-related expertise in complex environments". In J. A. Cannon-Bowers and E. Salas (Eds.), *Decision Making Under Stress: Implications for Individual and Team Training*. Washington, DC: American Psychological Association.

8. Lenox, T., Lewis, M. Roth, E., Shern, R., Roberts, L., Rafalski, T., and Jacobson, J. (1998). "Support of Teamwork in Human-Agent Teams." *IEEE Conference on Systems, Man, and Cybernetics*, 1341-1346.

9. Salas, E., Dickinson, T.L., Converse, S.A., and Tannenbaum, S.I. (1992). "Towards an understanding of team performance and training." In R. W. Swezey and E. Salas (Eds.), *Teams: Their training and performance*. Norwood, NJ:Ablex. 3-29.

10. Rouse, W.B., Cannon-Bowers, J.A. and Salas, E. (1992) "The role of mental models in team performance in complex systems." *IEEE Conference on Systems, Man, and Cybernetics*, 22, 1296-1308.

11. Oser, R.L., Jentsch, F.G., Bowers, C.A. and Salas, E. (1999). "Using automation to support decision making: challenges and training research requirements." *Proceedings of the Human Factors and Ergonomics Society 43rd Annual Meeting.*

12. Cannon-Bowers, J.A., Sallas, E., Blickensderfer, E. and Bowers, C.A. (1998). "The impact of cross-training and workload on team functioning: A replication and extension of initial findings." *Human Factors*, 40(1), 92-101.

13. Guzzo, R.A., and Dickson, M.A. (1996). "Teams in organizations: Recent research on performance and effectiveness." *Annual Review of Psychology*, 47, 307-338.