Varying the User Interaction within Multi-Agent Systems

Terry R. Payne
The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15232 USA
terryp@cs.cmu.edu

Katia Sycara
The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15232 USA
katia@cs.cmu.edu

Michael Lewis
School of Information Sciences
University of Pittsburgh
135 N. Bellefield Avenue
Pittsburgh, PA, 15260 USA
ml@lis.pitt.edu

ABSTRACT

Agents within an open multi-agent system are located through their advertisements with middle agents. Such advertisements describe the agent's capability, ontology, query or task specification, and details about the data returned once the task has been completed. Agents may have similar capabilities, but exhibit different models of user interaction. A case study of a multi-agent system is described which contains a variety of different agents, some of which have functionally similar capabilities but involve different types of user interaction. We demonstrate how the choice of user interaction can have a significant effect on the performance of the whole agent community. This leads to the proposal that an agent's interactive style should also be included within its capability advertisement.

General Terms

Middle Agents, Matchmakers, Task Agents, Interface Agents, Multi-Agent Teams, Collaboration, Human-Agent Interaction

1. INTRODUCTION

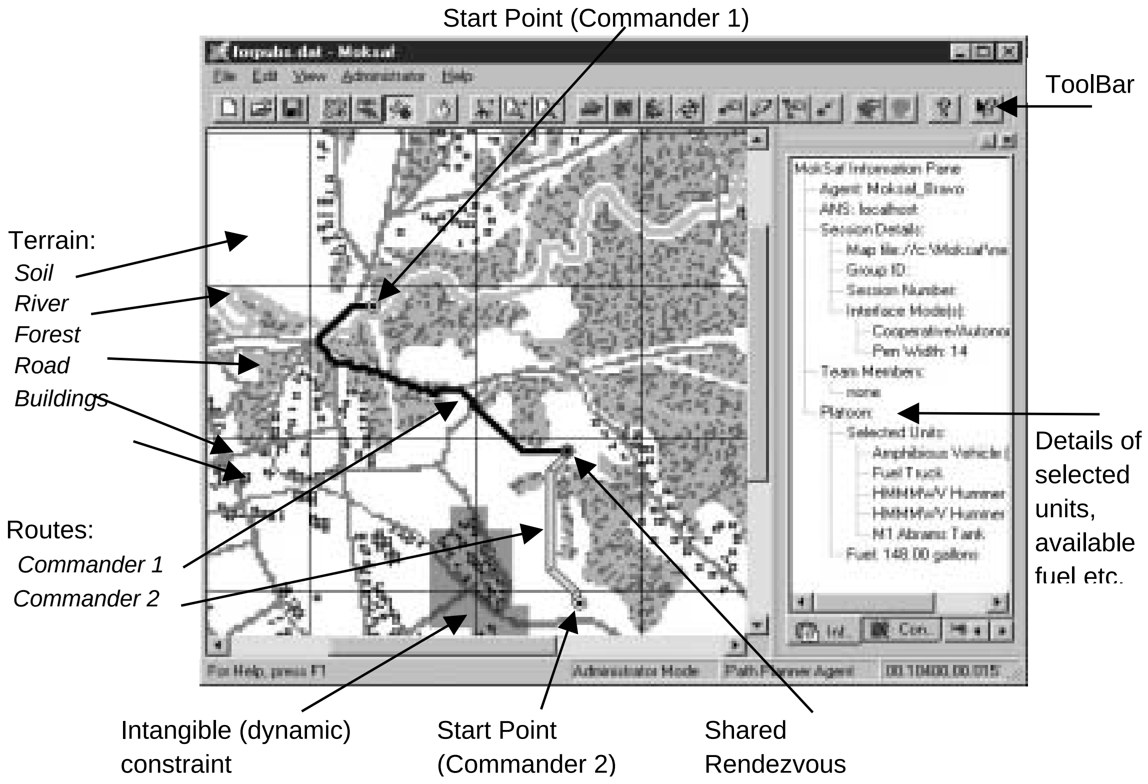
As the number and variety of agents that appear on the World Wide Web increase, frameworks that allow agents to seek and cooperate with other relevant agents have emerged. Various software agents that provide different services have already been deployed on the Web. These agents may provide specialized or periodic information from certain information sources, or may perform some task or service based on information they are given. From the perspective of services, there are three general agent categories: *service providing agents*, *service requesting agents* and *middle agents* that help service providers find service requesters [18]. Service providers possess certain know-how or *capabilities* (e.g. planning routes between two points within a city) that service requesters may desire. In addition, service providers have a set of service parameters (e.g. cost, reliability, availability etc.). Service requesters, on the other hand, have a set of *preferences* for particular types of parameters associated with desired capabilities. Providers advertise their capabilities and service parameters with one or more middle agents, and requesters request agents with particular capabilities and select a provider according to their preferences.

There are several different ways in which middle agents interact with service providers and service requesters [18], depending on factors such as reliability, privacy, efficiency etc. For example, middle agents may have knowledge of both the capabilities and preferences of different agents, and thus act as *brokers* [21]; such middle agents are often used within market-based systems. Another sub-class of middle agents are generally known as *matchmakers*, *yellow pages* or *directory agent* systems [18; 20; 5; 7]. These only have knowledge about the capabilities of service providers. Thus, if an agent has some preferences, it can query the middle agent, which then returns a list of agents whose capabilities match the preference query.

In addition to middle agents, other classes of agent exist. Agents may also be classed as *information agents*, *task agents*, and *interface agents*. In the Retsina system, for example [17], interface agents interact with the user, providing a mechanism whereby users can specify tasks, and inspect the results. They may acquire, model, and utilize user preferences to guide system coordination in support of the user's tasks [13]. Task agents help users perform tasks by formulating problem solving plans and carrying out these plans through querying and exchanging information with other software agents [12]. Information agents provide intelligent access to a heterogeneous collection of information sources [3].

The choice of task agent and the approach used by the interface agent to interact with the human can affect the behavior and utility of the agent society. Many interface agents determine user profiles to *personalize* the performance of an agent, or to improve the assistance provided by the agent [10; 11; 13]. In doing so, the interface agent may have to contact several different service providing agents, where each agent might share the same capabilities. The choice of agent tasked will not only affect the overall utility of the assistance provided by the interface agent, but may require the agent to change the way in which it interacts with the human.

In this paper, we examine the role of interaction between the human and interface agent, and how this interaction is determined by the choice of service provider. We demonstrate


Start Point (Commander 1)
ToolBar
Terrain:
Soil
River
Forest
Road
Buildings
Routes:
Commander 1
Commander 2
Details of selected units, available fuel etc.
Intangible (dynamic) constraint
Start Point (Commander 2)
Shared Rendezvous
File Edit View Administrator Help
Inf. Con.

MokSaf Information Pane
- Agent: Moksaf_Bravo
- ANS: localhost
- Session Details:
  - Map file://c:\Moksaf\ne
  - Group ID:
  - Session Number:
  - Interface Mode(s):
    - Cooperative/Autonor
    - Pen Width: 14
- Team Members:
  - none
- Platoon:
  - Selected Units:
    - Amphibious Vehicle (
    - Fuel Truck
    - HMMMWV Hummer
    - HMMMWV Hummer
    - M1 Abrams Tank
  - Fuel: 148.00 gallons

For Help, press F1 | Administrator Mode | Path Planner Agent | 00.10400.00.015

Figure 1: *MokSAF* Interface Agent.

that as the interaction varies, the utility of the interface agent can change. Thus, whist some agents may share similar advertisements (in terms of capabilities), the way they interact with the human through the interface agent can differ and thus have an effect on the overall utility. This is illustrated though the *MokSAF* multi-agent environment [14], which has been used to explore team dynamics, and to investigate the roles that different agents can play within a heterogeneous team of humans and software agents. Finally, we propose that the advertisement of an agent should include information about the mode of user interaction, and suggest that an ontology for interactions should be provided. Section 2 introduces the *MokSAF* environment, and describes the different *MokSAF* agents. Three different task agents that have similar capabilities but involve different user interactions are also presented. The experimental scenario and methodology are described in Section 3. The results are presented in Section 4 and the implication for agent advertisements are discussed in Section 5.

2. THE MOKSAF ENVIRONMENT

Emergency response tasks, both military and civilian, are characterized by environmental uncertainty, stress, and time criticality of decision making. The decision making process is distributed across different team members with different expertise, who are distributed in space and time, and who act with incomplete information in an uncertain environment. Hence, high quality computer assistance is critical. Software agents have emerged as a suitable metaphor for interacting with computer processes that assist human decision making. Such 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 [22]. In addition, such agents should not only retrieve information on request; but they should actively and intelligently anticipate, adapt and actively seek ways to support users [1; 19].

However, the benefits of software agents may be undermined by an increase in complexity and resulting confusion when interacting with the agent. New skills, such as task decomposition and delegation, may be required to interact with sophisticated software agents or agent communities [19]. 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 and leave us uncertain as to what is being done on our behalf [9]. It is important that users are able to construct their own goals and values, then decide, plan and act in ways to help them achieve goals and values. User autonomy can be reduced when users fail to understand what is happening within a system, when they cannot control the system, or when the agent and/or system behavior is unpredictable. Deskilling may also occur when the agent makes decisions for the user rather than just providing advice, or if the user is prevented from making his/her own decisions [6; 16]. These difficulties can be compounded where multiple agents and humans are required to work as a team. Under such conditions, cascading delegation among software agents and passive assistance may complicate the already challenging task of cooperating, communicating, and monitoring the other human team members.

The *MokSAF* environment is an open architecture multi-agent environment developed to explore teamwork within heterogeneous human/agent teams. It consists of a number of interface agents that present situation information to the human team members, provide tools to allow the team members to investigate a variety of different plans, and to communicate and coordinate with other humans through other interface agents. In addition, a variety of route planning agents are available within *MokSAF* that assist the humans when generating plans. These agents are not contacted directly, but tasked indirectly through each user's interface agent. A *Matchmaker* middle agent is used by the agents to register agent capabilities, and locate other agents. The agents use KQML [5] to communicate with each other, and with the middle agent.

2.1 MokSAF Interface Agents

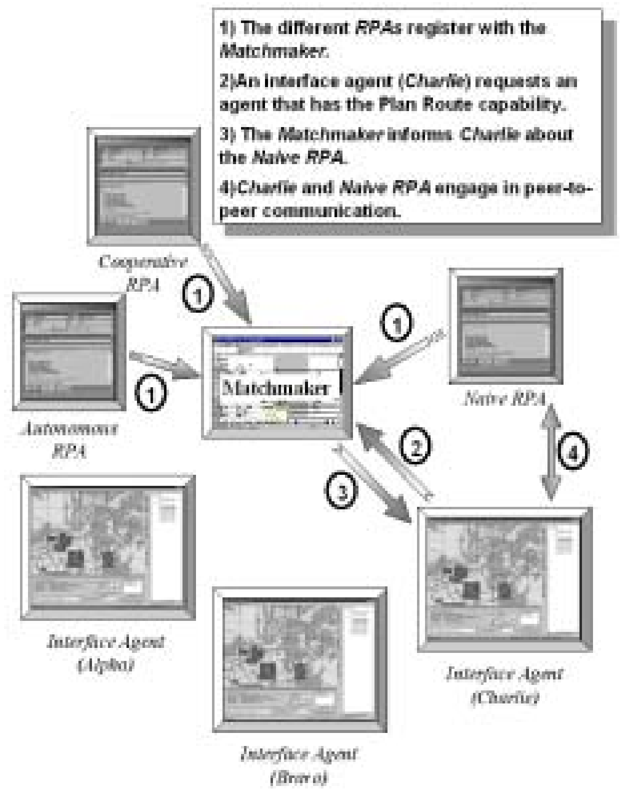
MokSAF interface agents provide a mechanism by which humans (acting as military commanders) collaborate in planning a military exercise or mission. A mission plan consists of one or more platoons of heterogeneous units, an agreed rendezvous time and location, and a set of routes for each platoon. The platoons start from different points, and each route ends at a common rendezvous. Each commander is responsible for the composition of one of the platoons, and for determining the route taken by that platoon. The commanders determine the individual routes and platoon compositions via the *MokSAF* interface agents and route planning agents or *RPA*s. In addition, the interface agents allow commanders to share routes with each other, and display the different routes on his/her interface agent (See Figure 1).

MokSAF is loosely based on a virtual battlefield simulation called MODSAF (MODular Semi-Automated Forces), and has been implemented within the Retsina multi-agent framework [17]. Although MODSAF is a rich simulation environment, the training and knowledge requirements are beyond what can easily be provided to the participants as part of this research. Figure 1 shows part of a *MokSAF* interface agent, including the terrain map and the toolbar.

When *MokSAF* is invoked, each commander's interface agent and several route planning agents register with the Matchmaker. Each interface agent displays a list of other agents that are registered and that can participate in a mission. A commander can then use the interface agent to contact other commanders and other *MokSAF* agents.

A commander can use the tools supplied by the interface agent to inspect the characteristics of a variety of different vehicles and other units, and to form the platoon. There are a number of factors the commander must consider, such as limited resources (e.g. fuel and unit types), the local geography (which may be unsuitable for certain units) and the speed characteristics of the units themselves. He/she can also inspect a terrain map, and may have to update the map to reflect mission data, such as locations that must be avoided, changes to the destination etc.

To provide effective assistance, a commander's interface agent needs to be aware of *intangible constraints*, which may be transient and unstructured, such as *"when on an exercise, avoid routes that go near schools during a semester, unless the platoon consists of light vehicles"*. These should be encoded in a form that can then be shared with other agents. Some form of feedback should also be provided so that the commander can verify that the encoding is correct. Intangible constraints are represented by shaded rectangles drawn on the terrain map, which represent areas that the platoon should avoid. Once drawn, these constraints can be shared with other agents such as the route planning agents, which can then utilize this knowledge when providing assistance with future routes.


1) The different *RPAs* register with the *Matchmaker.*

2) An interface agent (*Charlie*) requests an agent that has the Plan Route capability.

3) The *Matchmaker* informs *Charlie* about the *Naive RPA.*

4) *Charlie* and *Naive RPA* engage in peer-to-peer communication.

Figure 2: Registering with a Matchmaker.

Each commander can communicate with other commanders through their respective interface agents. These agents maintain a history of the communications for later reference. Each sent message is annotated with the name of the issuing commander. Commanders can elect whether to "broadcast" their messages to all the other commanders on the team or whether to send the message to a specific teammate. Messages consist of textual dialog pertaining to the commander's plans, negotiations regarding the allocation of units, recommendations for suggested changes to the rendezvous location and/or time, and other requests for information. In addition, commanders may also share their latest routes with other commanders. These shared routes can then be superimposed on the individual maps (as illustrated in Figure 1).

As the commanders formulate their plans, they may request assistance from a *route planning agent* or *RPA*. Three *RPA*s have been developed and are described in the following section. The mechanism used to register and locate agents is illustrated in Figure 2. As each *RPA* is created, it registers it's capabilities with the Matchmaker (1). For now, all agents within the *MokSAF* environment assume the same ontology to avoid the ontological mismatch problem [2]. Each time a commander requests assistance with generating a route, the interface agent contacts the Matchmaker and requests a list of agents that can provide this service (2). The Matchmaker compares the request with the capabilities that have been registered by the different *RPA*s, and returns this list to the interface agent (3). The interface agent can then directly contact any of the agents in the returned list (4), and request assistance with generating a route.

2.2 *M*okSAF Route Planning Agents

Three different methods for determining individual routes have been investigated using three different *RPA*s: the *Naive RPA*, *Cooperative RPA* and *Autonomous RPA*. These agents assist the commanders by helping with one aspect of the commander's task; planning a route. Thus the commander can focus on achieving the correct composition of units at the rendezvous.

Although several other studies have proposed various route-planning algorithms [8; 15], the agents described here utilize an off-road route-planning algorithm based on Dijkstra's shortest path algorithm [4] to generate or refine minimum cost routes between two points. Each *RPA* consults a digital terrain map that represents several classes of terrain (such as highways, freeways, forested areas, grassland, waterways etc.). The cost of traversing each pixel on the map is computed as the weighted sum of several coefficients, where each coefficient is dependant on the terrain type and two weighted platoon attributes: speed and fuel economy. The speed attribute is determined by finding the slowest speed of each of the units within that platoon (i.e. the platoon may travel no faster than its slowest unit). The fuel-economy attribute corresponds to the total volume of fuel consumed by the vehicles within the platoon. As certain units may only traverse certain terrain types, this may also constrain the route of the final path. Each terrain type has an associated traversal weight; these can be used to favor or discourage routes from crossing certain terrain. The attribute weights determine the relevance of the two platoon attributes (speed and fuel-economy). Thus, by varying these weights, it is possible to determine the fastest route, the most fuel economic route, or a route that combines both attributes.

Naive RPA

The *Naive RPA* acts more as an adviser or critic, than as an agent that generates routes. When tasked, it requests information about an existing route, the characteristics of the platoon, and details of any existing intangible constraints. The route consists of a series of midpoints between a start and rendezvous point. The commander is responsible for plotting these midpoints on the terrain map within his/her interface agent (Figure 3). The *Naive RPA* then analyses this route, and checks it's validity. A valid route is one that avoids intangible constraints, and that crosses those terrains suitable for the platoon (e.g. it checks that a platoon comprising of heavy tanks does not attempt to cross a large river or dense forest). If such violations occur, the route is annotated to reflect this. In addition, estimates of fuel consumption and arrival time are calculated. If the platoon runs out of fuel en-route, the route is also annotated. This information is then returned to the interface agent, which presents the annotated map to the commander.

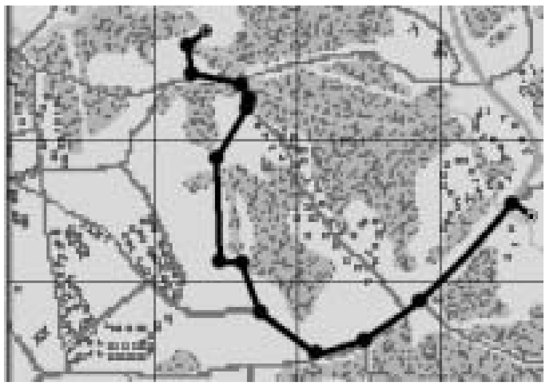


Figure 3: When tasking the *Naive RPA*, the interface agent represents routes drawn by the commander as line segments between a series of midpoints on the map.

As the *Naive RPA* provides very simple assistance to the commanders, it has been used as a baseline from which to evaluate the performance of the other route planning agents.

Cooperative RPA

The *Cooperative RPA* is similar to the *Naive RPA* as it also requests a series of midpoints between a start and rendezvous point, and details about the platoon and intangible constraints. However, in this case, the midpoints do not represent a specific route, but represent a wide corridor which constrains the commander's desired route (see Figure 4). The *Cooperative RPA* generates a new route within this corridor, that maximizes the speed of the platoon and thus reduces the expected time of arrival of the platoon. This route takes into account the characteristics of the platoon, and avoids crossing intangible constraints.

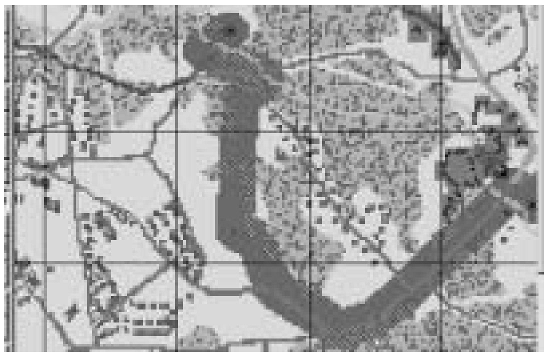


Figure 4: When tasking the *Cooperative RPA*, the interface agent represents routes drawn by the commander as corridors across the map. The *Cooperative RPA* then determines a route within this corridor.

There is no guarantee that if the platoon follows the route, it will arrive at the rendezvous point with fuel to spare. Thus, the agent estimates the fuel costs, and if there is a shortage of fuel, the route is annotated to reflect this.

Autonomous RPA

The *Autonomous RPA* differs slightly from the previous two agents in that it discards the midpoints generated by the commander. Instead, it determines the optimal route between the start and rendezvous points, taking into account the characteristics of the platoon, and any intangible constraints that the commander has defined. If estimates of fuel usage indicate that the platoon will run out of fuel en-route, then the attribute weights are adjusted until a more appropriate route is found. If no such route can be generated, then the fastest route will be annotated to indicate at what point fuel levels are exhausted, and this annotated route is sent to the interface agent for presentation to the commander.

The commander may request alternative routes by modifying the weights associated with each type of terrain. The commander specifies whether or not a specific type of terrain should be preferred, and the corresponding weight is then modified until a new route is generated [15]. Alternatively, the commander can request a faster or more economic route.

3. EXPERIMENTATION

We conducted three sets of experiments to test the effectiveness of the assistance that the different *RPA*s and their interaction modes can provide. Twenty-five three-person teams were recruited from the University of Pittsburgh and Carnegie Mellon University communities. Subjects were recruited as intact teams, consisting of friends or acquaintances. Teammates needed to communicate with one another to complete their tasks successfully. Within each experiment, only one of the three route planning agents were made available to the commanders. Thus, the assistance of each type of *RPA* can be compared and contrasted. The *Autonomous RPA* used in this study used static weights, and thus commanders were unable to request alternative routes by modifying terrain or attribute weights.

3.1 Experimental Scenario

The problem that the team of commanders has to solve within an experimental scenario is as follows: each commander must select appropriate vehicles to constitute his/her platoon so that:

1. The platoon should reach the shared rendezvous point before a given deadline.
2. The platoon should reach the shared rendezvous point without running out of fuel.
3. The route taken by each platoon should result in minimal possible fuel consumption.
4. The platoon should visit certain mid-points en-route.
5. The route should not violate any physical constraints (such as crossing densely forested areas with large vehicles).
6. The route should not violate intangible constraints (where an intangible constraint might specify *"avoid this region as it is a suspected minefield"*).
7. The combined platoons should contain a minimum subset of specified units at the rendezvous.

This is a complex constraint optimization problem. Each vehicle has different characteristics with respect to the types of terrain it can traverse and moreover, the vehicle's speed and fuel consumption depends on the type of terrain being crossed at a given time interval. In addition, the intangible constraints must be somehow represented so they can be taken into consideration during problem solving. Finally, the problem is of large scale since the planned route can be off-road i.e. vehicles traverse open spaces, such as desert, grassy areas, or forests (without the advantage of having marked roads to constrain the search).

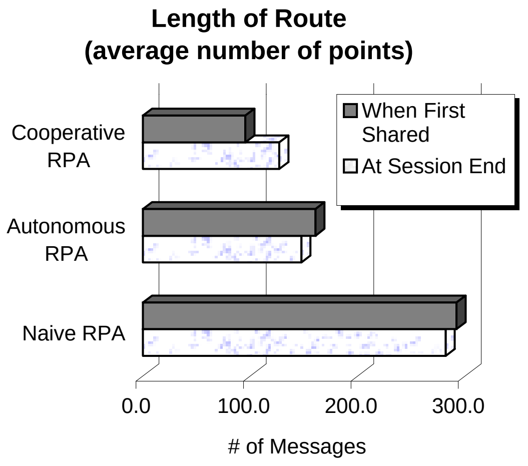
This task has a variety of characteristics, some of which are easy for humans to deal with and some that are difficult. What makes the task easy for humans is its visual nature, namely the fact that routes can be drawn on the map. In contrast, it is very difficult for humans to calculate path lengths, vehicle speeds or fuel consumption. These latter characteristics make the task more amenable to agent-based team assistance.

Each team member was assigned the role of one of three commanders (*Alpha*, *Bravo* or *Charlie*). Each commander was given a different starting point but shared a common rendezvous point with the others. The commanders were told to coordinate the number and types of units that they planned to move from the individual start points to the rendezvous point. A mission briefing was supplied to the commanders which listed a minimal subset of vehicles that should be at the rendezvous point at the end of each session. In addition, the commanders were instructed to avoid generating routes that lay on the same path as those generated by the other commanders, and that they should coordinate their routes to avoid this. Each commander selected units for his/her platoon from a list of available units. Commanders were given 15 minutes to determine the composition of their platoon, and plan a route from a starting point to the rendezvous point for that platoon. Once a commander was satisfied with the individual route, this route should be shared with the other commanders to identify and resolve any conflicts. Conflicts could arise due to shared routes, shared resources, or the inability of a commander to reach the rendezvous point at the specified time.

4. RESULTS

In general, the routes generated with the assistance of the *Naive RPA* were found to be longer, and hence platoons consumed more fuel and arrived later at the rendezvous than the other two *RPA*s. The route lengths were measured at two time intervals; when each commander first shared his/her individual route with the other commanders, and at the end of the session. Thus the performance of each commander could be evaluated when the individual routes are first generated (and hence reflect the commander's individual goals), and when the routes (and platoon composition) are refined in light of the other commander's routes.

The bar graphs in Figure 5 illustrate the differences in route lengths when comparing all three *RPA*s. The routes generated by the *Autonomous RPA* or *Cooperative RPA* were significantly shorter ($p < 0.001$) than the *Naive RPA*. Although there was no significant difference between the routes generated by either *Autonomous RPA* or *Cooperative RPA*, commanders took longer in generating routes prior to sharing when interacting with the *Cooperative RPA* than the users who were assisted by the *Autonomous RPA*, with more than a third of them failing to find a valid route before their team mates initiated route sharing. Despite this, *Cooperative RPA* users were found to identify shorter paths ($p < 0.01$) than *Autonomous RPA* users at time of route sharing and were found to maintain a slight (though not statistically significant) advantage by the end of the session (see Figure 5).



	Naive RPA	Autonomous RPA	Cooperative RPA
When First Shared	292.0	160.6	95.3
At Session End	281.8	147.5	127.0

Figure 5: Variation in the length of the generated routes.

The most observable difference between the three route planning agents was not in the quality of the routes generated, but rather in the substantially greater time that it took users of the *Naive RPA* to construct a route. The coordinated routes at the end of each experiment were uniformly better for each of the commanders that interacted with the *Autonomous RPA* or the *Collaborative RPA*, and also for the team as a whole. Despite this clear superiority, subjects who used the *Autonomous RPA* frequently expressed frustration with the indirection required to coerce the behavior of the agent by adding additional intangible constraints to the map, and then requesting new routes. Several subjects remarked that they wished they could "just draw the route by hand".

With the *Naive RPA*, subject complaints focused more closely on the minutiae of interaction. In its current form, the user *"draws"* a route within the interface agent. Subjects commented that this process was both tedious and error prone. When commanders constructed routes, low fidelity routes frequently violated terrain constraints (such as passing through buildings), and failed to follow optimal paths such as curves in roads. Although the inclusion of additional points into the route was found to resolve this problem, the process was time consuming. Hence, subjects found that they spent more time refining their individual routes, as opposed to coordinating with other team members to improve the overall team task.

Subjects who used the *Cooperative RPA* seemed satisfied with their modes of interaction, and yet experienced some difficulty in rapidly constructing valid routes. This difficulty was particularly acute for the *Alpha* commanders. This suggests that neither the *Cooperative RPA* or *Autonomous RPA* provide a panacea for route planning, but that each has it's own advantages and disadvantages, depending on the type of route required.

5. DISCUSSION

We have developed a complex team planning environment in which users interact with various interface agents, and can task other agents (in this case, one of three route planning agents) to assist them in achieving their goals. Although three very different *RPA*s were developed, they all had similar capabilities, each requested the same input parameters, and returned the same type of output parameters. However, each required different types of interaction with the user. The experiments demonstrated that of the three, the *Cooperative RPA* and *Autonomous RPA* both generated superior routes. In addition, although the routes generated *Cooperative RPA* were slightly better, users could generate routes quicker when using the *Autonomous RPA*.

Although a matchmaker was used in the above experiments, in each case only a single type of *RPA* was registered. In our current implementation, the advertisements of the three route planning agents can only be differentiated through idiosyncratic labels within the service parameters; i.e. there is a service parameter which has one of three enumerated types: {*naive, autonomous, cooperative*}. This parameter is also used by the interface agent to determine the type of interaction to present to the user. Whist this has provided a short term ad-hoc solution to the problem of differentiating between the *RPA*s, it is clearly not ideal. Such an approach might be acceptable within a closed world agent community. However, these labels are meaningless within an open agent community.

Some description of the user interaction is necessary not only to differentiate the different agents, but also to define the type of interaction in which the interface agent should engage the user. One possibility would be to describe the necessary mode of interaction that an interface agent should present as a finite state machine. However, such a description would be impractical for all but the simplest agents, and the formalism would be very difficult to both define and prove.

We are currently exploring the feasibility of proposing a new ontology of user interactions. Such an ontology might consist of a taxonomy of interaction types, ranging from higher level abstract interactions to lower level atomic interface widgets. Thus, future interface agents would be able to locate task agents, and from the advertisement, construct new interface modes based on this new ontology which could then be presented to the user.

6. ACKNOWLEDGEMENTS

This research was supported by an Office of Naval Research Grant N-00014-96-1-1222. Special thanks go to C. Domashnev & M. van Velsen for their assistance in the development of this software.

7. ADDITIONAL AUTHORS

Additional authors: Terri L. Lenox (email: `tll@lis.pitt.edu`) and Susan Hahn (email: `hahns@lis.pitt.edu`).

8. REFERENCES

[1] J. Bradshaw. Introduction. In J. Bradshaw, editor, *Software Agents*, pages 3–48. AAAI Press:Menlo Park, CA, 1997.

[2] C. Collet, M. Huhns, and W. Shen. Resource Integration using a large knowledge base in Carnot. *Computer*, 24(12):55–62, 1991. December.

[3] K. Decker, A. Pannu, K. Sycara, and M. Williamson. Designing Behaviors for Information Agents. In *Proceedings of the First International Conference on Autonomous Agents (Agents-97)*, pages 404–412. ACM Press: New York, 1997.

[4] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–318, 1959.

[5] T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an agent communication language. In *Proceedings of the third Conference on Information and Knowledge Management, CIKM94*. ACM Press: New York, 1994.

[6] B. Friedman and H. Nissenbaum. Software agents and user autonomy. In *Proceedings of the First International Conference on Autonomous Agents (Agents '97)*, pages 466–469. ACM Press: New York, 1997.

[7] M. Genesereth and S. Katchpel. Software Agents. *Communications of the ACM*, 37(7):48–53, 1994.

[8] K. Haigh, J. Shewchuk, and M. Veloso. Exploiting Domain Geometry in Analogical Route Planning. *Journal of Experimental and Theoretical AI*, 9:509–541, 1997.

[9] M. Lewis. Designing for human-agent interaction. *AI Magazine*, pages 67–78, 1998. Summer.

[10] P. Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):31–40, 1994.

[11] D. Maulsby and I. Witten. Teaching Agents to Learn: From User Study to Implementation. *Computer*, 30(11):36–44, 1997.

[12] M. Paolucci, D. Kalp, A. S. Pannu, O. Shehory, and K. Sycara. A planning component for retsina agents. In *Lecture Notes in Artificial Intelligence, Intelligent Agents VI*, 1999.

[13] T. Payne, P. Edwards, and C. L. Green. Experience with rule induction and k-nearest neighbour methods for interface agents that learn. *IEEE Transactions on Knowledge and Data Engineering*, 9(2):329–335, March 1997.

[14] T. Payne, T. Lenox, S. Hahn, M. Lewis, and K. Sycara. Agent-Based Team Aiding in a Time Critical Task. In *Proceedings of the Thirty-Third Hawai'i International Conference on System Sciences (HICSS33)*, 2000.

[15] S. Rogers, C.-N. Fiechter, and P. Langley. An Adaptive Interactive Agent for Route Advice. In *Proceedings of the Third International Conference on Autonomous Agents (Agents-99)*, pages 198–205. ACM Press: New York, 1999.

[16] N. B. Sarter and D. Woods. From tool to agent: the evolution of (cockpit) automation and its impact on human-machine coordination. In *Proceedings of the Human Factors and Ergonomics Society 39th Annual Meeting*, pages 79–83, 1995.

[17] K. Sycara, K. Decker, A. S. Pannu, M. Williamson, and D. Zeng. Distributed intelligent agents. *IEEE Expert*, 11(6):36–46, December 1996.

[18] K. Sycara, K. Decker, and M. Williamson. Middle-agents for the internet. In *Proceedings of IJCAI-97*, January 1997.

[19] K. Sycara, K. Decker, and D. Zeng. Intelligent Agents in Portfolio Management. In N. Jennings and M. Wooldridge, editors, *Agent Technology: Foundations, Applications, and Markets*, pages 267–283. Springer, 1998. Chapter 14.

[20] K. Sycara, M. Klusch, S. Widoff, and J. Lu. Dynamic service matchmaking among agents in open information environments. *SIGMOD Record (ACM Special Interests Group on Management of Data)*, 28(1):47–53, March 1999.

[21] M. Wellman. A Market-Oriented Programming Environment and its Application to Distributed Multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1:1–23, 1993.

[22] W. Zachary, J.-C. Le Mentec, and J. Ryder. Interface agents in complex systems. In C. Ntuen and E. Parks, editors, *Human Interaction with Complex Systems: Conceptual Principles and Design Practice*, pages 267–283. Kluwer Academic Publishers, 1996. Chapter 14.