User evaluation of a GUI for controlling an autonomous persistent surveillance team

Paul Scerri[a], Sean Owens[b], Katia Sycara[c] and Michael Lewis[d]

[a]Robotics Institute, Carnegie Mellon University, Pittsburgh, USA;
[b]Robotics Institute, Carnegie Mellon University, Pittsburgh, USA;
[c]Robotics Institute, Carnegie Mellon University, Pittsburgh, USA;
[d]School of Information Sciences, University of Pittsburgh, USA.

ABSTRACT

In future military missions, there will be many sensor assets collecting much important information about the environment. User control over surveillance assets is important to ensure that the specific data collected is appropriate for the current mission. Unfortunately, previous work has shown that individual users cannot effectively control more than about four assets, even if the assets have significant autonomy. In the ACCAST project, we hypothesized that by including autonomous teamwork between the assets and allowing users to interact by describing what the team as a whole and specific sub-teams should do, we could dramatically scale up the number of assets an individual user could effectively control. In this paper, we present the results of an experiment where users controlled up to 30 autonomous assets performing a complex mission. The assets autonomously worked together using sophisticated teamwork and the user could tell sub-teams to execute *team oriented plans* which described the steps required to achieve a team objective without describing exactly which asset performed which role and without having to specify how the team should handle routine information sharing, communications and failure circumstances. The users, soldiers from Fort Benning, were surprisingly good at managing the assets and were *all* able to complete the complex mission with extremely low friendly and civilian casualties.

Keywords: persistent surveillance, user interaction, teamwork

1. INTRODUCTION

It is projected that a deployed Brigade Combat Team in the coming decade could employ hundreds of Unmanned Aerial Vehicles (UAVs) simultaneously in support of a broad range of mission functions from ISR, to communications, to actual weapons deployment. Similar projections illustrate even more Unmanned Ground Vehicles (UGVs) and support vehicles. As these missions increase in complexity, the need for coordination and control among and across teams of UAVs, manned platforms, control systems, and human elements increase exponentially. However, successful employment is critically dependent upon effective methods for their coordination and control in an environment fraught with communications barriers and complications. For example, increased deployment within urban environments leads to line-of-sight communications disruptions, intentional (electronics countermeasures by adversaries) and unintentional (e.g. urban electro-magnetic interference) noise, as well as the simple competition among friendly forces for available bandwidth.

The enormous increase in unmanned systems, coupled with the fact that unmanned does not equate to unattended results in a huge increase in operator workload. This leads to decreased situation awareness that can translate to greatly reduced mission effectiveness. Current unmanned systems are heavily manpower dependent. Miller, for example, showed that under expected target densities, an Unmanned Combat Air Vehicle (UCAV) controller who is required to authorize weapon release could control no more than 13 Unmanned Air Vehicles (UAVs) even in the absence of other tasks.[1] Recent AFRL studies target an even more modest 4 UAVs/operator. Similar numbers (3-9)[2] have been found for Unmanned Ground Vehicles (UGVs). While we have removed the man from the vehicle, we have not automated enough systems to allow the units to work independently. As

Further author information: Paul Scerri: E-mail: pscerri@@cs.cmu.edu, Telephone: 1 412 268 2145

a result, current systems typically require two or more operators to control each UAV/UGV unit, either in a tele-operated mode, or in a command and control planning and monitoring configuration. Research is currently being performed to find ways to reverse this one-to-two-plus ratio of UAV/UGV-to-operators and achieve an N-to-1 ratio.[3–6] Such an achievement would allow one operator to control a fleet of unmanned assets.

The best way to achieve this objective is to put into place effective command and control algorithms that will allow the UAV/UGV/Unattended Ground Sensors/Mines (UGS) systems to team automatically, taking care of the coordination aspects and freeing the operator to focus on the higher-level tactical and strategic goals of the mission(s) being undertaken. In fact, the objective is to change the nature of the operator himself to that of a manager. The coordination problem will be exacerbated as the control of unmanned systems is pushed from rear echelon support units to the front line infantryman. Effective command and control mechanisms will free manpower resources from commitment to the dull, dirty, or dangerous warfare activities and instead focus their use in operations that are better suited to humans. This achievement will enable the UAV/UGV/UGS team to blossom into the force multiplier that has been the dream of operations planners since the inception of the Unmanned System concept.

We developed an Advanced Command and Control System for Teams (ACCAST) to implement this idea. ACCAST uses the Machinetta coordination software[7] to autonomously coordinate the unmanned assets. Machinetta provides all the intelligent reasoning required for a group of unmanned assets to work together, including facilitating information sharing, allocating tasks and monitoring for failures of individuals. The user interacts with the team via *team-oriented plans* which abstractly describe a team goal and a plan for achieving that goal, leaving the runtime software to handle all the details. The team plans are initiated and monitored via interface built on top of FalconView.[8] A team plan meta-description describes the possible team plans and any ways the plans can be configured. The FalconView plan interface takes the plan meta-description and generates dialog boxes for initiating and configuring the plans. We evaluated the system with 12 special forces trainees at Fort Benning in June 2009. The experiments showed that with minimal training the users were able to effectively control a team of up to 30 autonomous assets to achieve a complex mission. The users were able to perform ISR and locate nearly all opposing forces, before attacking targets, sending in manned vehicles and protecting them from new threats.

In the remainder of this paper, we describe the overall project, the interface developed and describe the results from the experiments in detail.

2. AUTONOMOUS COORDINATION INFRASTRUCTURE

The ACCAST project was aimed at allowing a single operator to control many ISR assets to achieve a tactical objective. The basic hypothesis was that autonomous coordination was key to relieving the user from the cognitive and physical burden of routine activity and that new interaction designs were necessary to leverage the infrastructure. In this section, we describe the coordination infrastructure used to autonomously coordinate the sensors and in the following section we describe the interface concept.

The ACCAST system is based upon Machinetta, which encapsulates coordination reasoning in a module that can quickly be interfaced to any robot or agent. Any system with a Machinetta software component can dynamically join a team of robots that also have Machinetta components on board and be utilized towards team goals. Because of this, the system has the ability to dynamically utilize unmanned systems as they become available to the team. Additionally, new classes of unmanned systems can be be quickly integrated into the team. The system is robust to unmanned systems being removed from the team, either due to attrition or being reassigned to another team.

The basis of coordination in the Machinetta proxies is a Team Oriented Plans (TOP). A TOP describes the joint activities that must take place for the team to achieve its goals. At any point in time, the team may be executing a number of TOPs simultaneously. TOPs are instantiated from TOP templates. These templates are designed before the team begins operation, typically by humans to ensure compliance with established doctrine or best practices. A TOP is a tree structure, where leaf nodes are called roles and are intended to be performed by a single team member. For example, a typical TOP for a UAV domain is to destroy a ground based target. Such a plan is instantiated when a ground-based target is detected. The plan is terminated when the target is

confirmed as destroyed or the target becomes irrelevant. The plan specifies that the roles are to actually hit the target and to perform battle damage assessment. The battle damage assessment must be performed after the target has been hit. The coordination algorithms built into the proxies,[9,10] handle the execution of the TOP, hence the plan does not describe the required coordination nor how the coordination needs to be performed. Instead the TOP describes the high level activities and the relationship and constraints between those activities.

A PlanAgent is responsible for managing a plan. This involves instantiating and terminating roles as required and stopping execution of the plan when the plan either succeeds, becomes irrelevant, or is no longer achievable. These conditions are observed in the proxy state. Currently, the PlanAgent must simply match conditions using string matching against post-conditions in the template, but one can envision more sophisticated reasoning in the future. Because plans are instantiated in a distributed manner, the PlanAgents need to ensure that there are not other plans that are attempting to achieve the same goal (e.g., hit the same target) or other plans that may conflict. To facilitate the conflict avoidance (and detection) process, as well as keeping the team apprised of ongoing activities, the first thing a PlanAgent does is create an InformationAgent to inform the other proxies. If the PlanAgent does not detect any conflicts, it executes its main control loop until the plan becomes either irrelevant, unachievable or is completed. For each role in the plan, a RoleAgent is created. RoleAgents are coordination agents that are responsible for a specific role. The RoleAgent is responsible for finding a team member to execute that role. As the plan progresses, the required roles may change, in which case the PlanAgent must terminate the current RoleAgents and create new RoleAgents for the new roles. It is also possible that a previously undetected plan conflict is found and one plan needs to be terminated. The PlanAgents responsible for the conflicting plans jointly determine which plan to terminate. When the plan is completed, the PlanAgent terminates any remaining RoleAgents and finishes.

ACCAST provides a distributed coordination system for UAVs, UGVs, UGSs and UMs that is robust to variations in available communications including the ability to gracefully degrade coordination when no communication is available for a period of time. The idea is to leverage a Lockheed Martin ATL developed communication wrapper with the ability to prioritize the sending of messages between vehicles to ensure that, when communication is impaired rather than completely unavailable, or when it is re-established after a blackout, the most critical messages get through, thus allowing coordination to proceed though in a gracefully degraded manner. This prioritization has the additional effect of reducing the amount of communication required by Machinetta, thus making coordination even more effective in a restricted-bandwidth or intermittent-communication situation. ACCAST reasons about and determine the importance of each coordination message class to the performance of a team. This allows coordination messages of higher importance to have precedence on the network. The process of determining the importance of a coordination message to the team requires an automatic process of reflection and an explicit representation of the purpose of that coordination message and the cost in the potential loss or delay of that message.

In any platform, the ACCAST system maintains two types of models. Firstly, it maintains a model of the current state of the coordination. Specifically, this model includes which Team Oriented Plans are being executed and the state of those plans. In a dynamic and distributed environment, each platform will not necessarily have a completely up-to-date picture of this state. Secondly, each platform maintains a model of the state of other platforms in the system, including their geographic location, their current activities and the ability to communicate with that platform. This model is used in concert with the coordination model to determine precisely which platform to inform about precisely which events. For example, it is intuitively clear that it is more important to pass information about detected SAM sites to UAVs likely to be in the vicinity of those sites. Moreover, the less constrained a UAVs communication, the wider a field of awareness regarding SAM sites might be relayed to that UAV.

We developed our own simulation environment, called Sanjaya, capable of running much faster than real-time to allow many simulation runs to be performed, while maintaining medium fidelity sensor and motion models.[11] The scenario used in these experiments requires that a team of about 30 unmanned assets take an airfield where a number of opposing forces and civilians are located. The user will typically perform ISR first, to find opponents, strike opposing forces that are found then move manned vehicles in to secure the area. The sensor models in Sanjaya accurately reflect error probabilities for real sensors. The uncertainty associated with a detection is shown to the user who may require additional ISR before striking the target. Effectiveness of strikes and

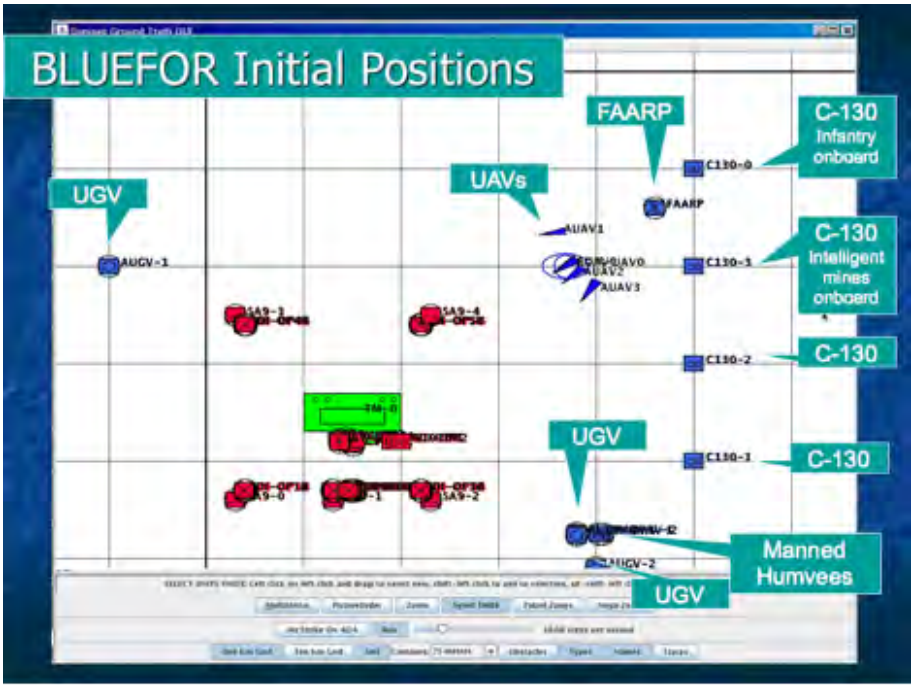

Figure 1. Sanjaya simulation environment with starting locations of friendly forces in the experimental scenario marked. The friendly forces start in the North East corner, with the exception of one UGV. The task is to take and hold the airfield in the center of the map. The friendly force is a mixture of unmanned air and ground assets, managed at a Forward Area Arming and Refueling Point (FAARP), where the commander is assumed to be based.

capabilities of various unmanned assets are representative of the types of capabilities expected to be available in the next 5-10 years.

3. INTERFACE CONCEPT

In this section, we describe the high-level design decisions and philosophy behind the ACCAST interface and in the next section we describe in more detail the specific interface elements. An important underlying assumption to these ideas is that the user interacting with the interface will have some tactical control. It is clearly not desirable to have senior commanders stuck sitting in front of computer screens, but as missions involve a bigger proportion of unmanned assets, neither will it be sensible to have very junior soldiers in charge of many assets, isolated from commanders. As missions rely more on unmanned assets doing the right thing, the commanders responsible for making the right tactical decisions need to be more in command of the autonomous assets. With this future in mind, our interface design is aimed at users with considerable tactical control over a mission.

One of the key ideas in ACCAST is that the operator should interact via a map-based interface that they are comfortable with and not with specific interfaces for specific sensors. Interacting via a map-based display allows the operator to think tactically about the evolving situation and not become too focused on what a particular sensor is doing. The map displays the tactical situation and allows the user to use the sensors as a tool to understand and shape the tactical situation instead of the sensor management being the primary concern.

Another key concept is that real-time data from sensors will be displayed in a way that is abstracted away from the particular sensor producing the data. This is particularly important when some information is due to fusion of data from multiple sensors, but even when the pedigree of the information is clear, it is our hypothesis that users should not be thinking about which sensor produced which information but simply be concentrating on the impact of the information on the tactical situation. This is in line with traditional military practice, where commanders are presented summaries of the situation, not raw data. Clearly, it is important for users to be able to determine where information came from, when its veracity is in doubt or when there may be considerable

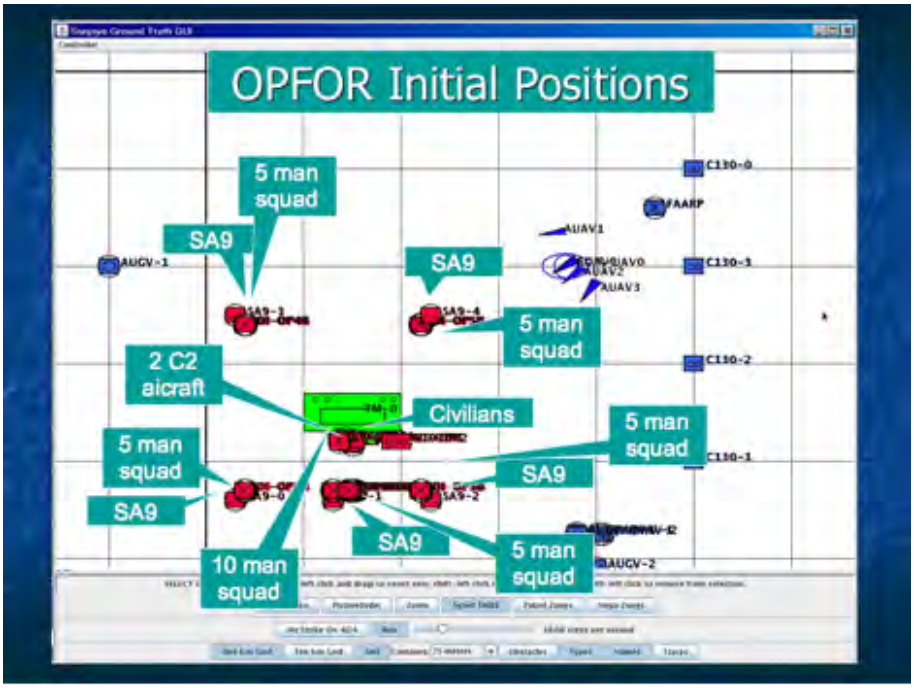

Figure 2. Sanjaya simulation environment with starting locations of opposing forces marked. The opposing forces are clustered around the airfield at the center of the map, interspersed with civilians, making accurate ISR important.

uncertainty. However, by default we believe the origin of the information should be abstracted away in the interface.

As noted above, we hypothesize that to increase the number of autonomous assets a single user can effectively control, there needs to be autonomous coordination between the autonomous assets and that interaction with the assets has to be done at a team level. Specifically, we are investigating how an autonomous team that manages its behavior via *team-oriented plans* reduces the load on the operator. Interaction with this team is done by starting and stopping team plans. Thus, a key interface concept is that a primary interaction mode will be via team plans. Specifically, during initialization, Machinetta communicates to the interface a specification of the set of plans the team can perform and the ways those plans can be configured. For example, one team plan might be to locate any targets in a particular area and that plan might allow the user to specify whether ground, air or a mixture of sensors can be used for the reconnaissance.

4. FALCONVIEW INTERFACE

FalconView[12] was chosen as a GUI platform because of its familiarity to many operators. It is a fairly standard and well-known package for vehicle monitoring and high-level control. ACCAST is integrated into FalconView via standard FalconView methodologies. FalconView provides APIs with which menu options can be inserted, and ACCAST inserts menu options to connect to ACsCAST, and to instantiate a plan. The instantiation of a plan is achieved by a sequence of pop-up windows, similar to a Wizard. Once instantiated, the plan is forwarded to Machinetta via a mechanism described in the next section.

The ACCAST FalconView component also watches the scenario unfolding and updates its map accordingly. Users may then get detailed information about onscreen units by clicking them. This information represents Machinettas beliefs, and is not necessarily fully specified or ground truth. Its simply whats known at the time, which provides the user with an accurate simulated experience.

Consistent with the ACCAST philosophy, the primary user interaction with the ACCAST FalconView component is for the users to instantiate plans. The plans that are available for instantiation are determined by

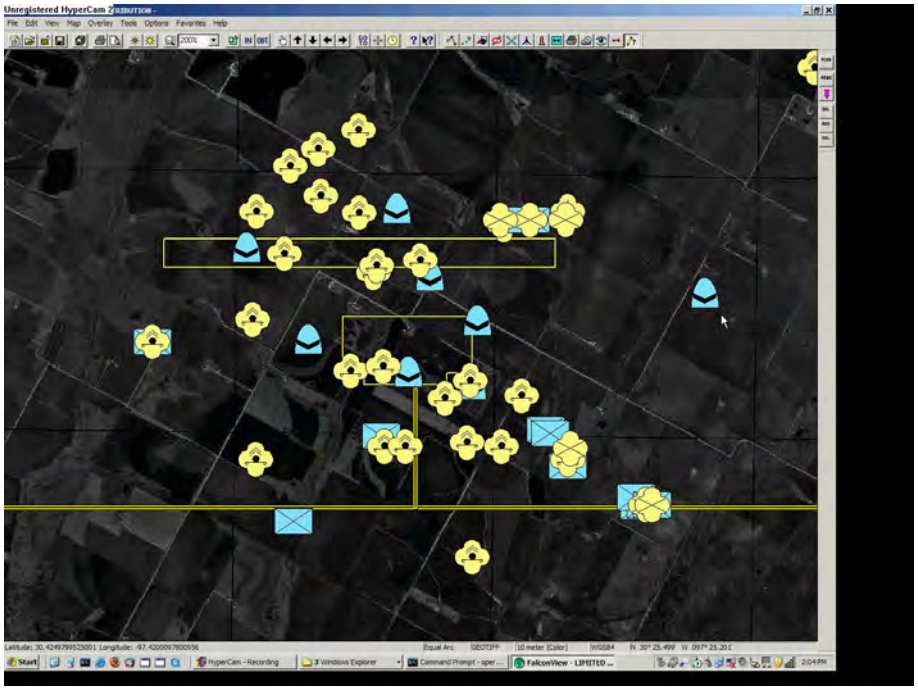

Figure 3. FalconView interface with sensor reports showing asset locations shown.

Machinetta; they are not programmed into the FalconView component, but rather are received by FalconView from Machinetta at runtime. Each plan has a number of fields, each of which has a Type. The ACCAST FalconView component knows how to ask the user for each type. Some are numbers, some are strings, some are locations on the map, and some are areas of the map in each case, the ACCAST FalconView component presents the user with a widget specifically designed to handle that type of input, including allowing the user to interact with the map. Thus, the ACCAST FalconView component is agnostic of specific plans, and plans may be added to Machinetta without modifying the FalconView component.

Figures 3, 4 and 5 show the resulting interface. Figure 3 shows the basic interface with a number of ISR reports shown. Figure 4 shows the user selecting a plan from a dialog box. In the background, information about friendly, civilian and opposing forces is displayed on the map. The user has the map in satellite mode, but they can switch to elevation, terrain or other map types as FalconView allows. Figure 5 shows the user filling out details of a plan they have selected.

5. RESULTS

Six groups of two users were given a two hour introduction to the ACCAST concept, a short familiarization session with the interface and then left to attempt to take the airfield. Groups of two users were used purely because time of the soldiers was very limited and we wanted as much feedback as possible. Usually the users shared control of the interface, though one pair split responsibilities so that one person controlled and the other observed and made key technical decisions. One log-file was damaged and did not allow processing, so is excluded from below, although we informally observed the outcome to be qualitatively the same.

Each of the five groups of two users successfully took the airfield. This was somewhat unexpected since the scenario was intentionally made difficult and the authors, who knew the scenario very well, were not always successful. Figures 6 - 10 show timelines of the plans initiated by each of the user groups. Notice that the groups approached the scenario very differently, with some creating many small ISR plans, interspersed with attacks, while others created a small number (as small as 1) of ISR plans then monitored data to order attacks. In most cases, the "other" plan is transport of soldiers in manned vehicles, though one user chose to airdrop soldiers

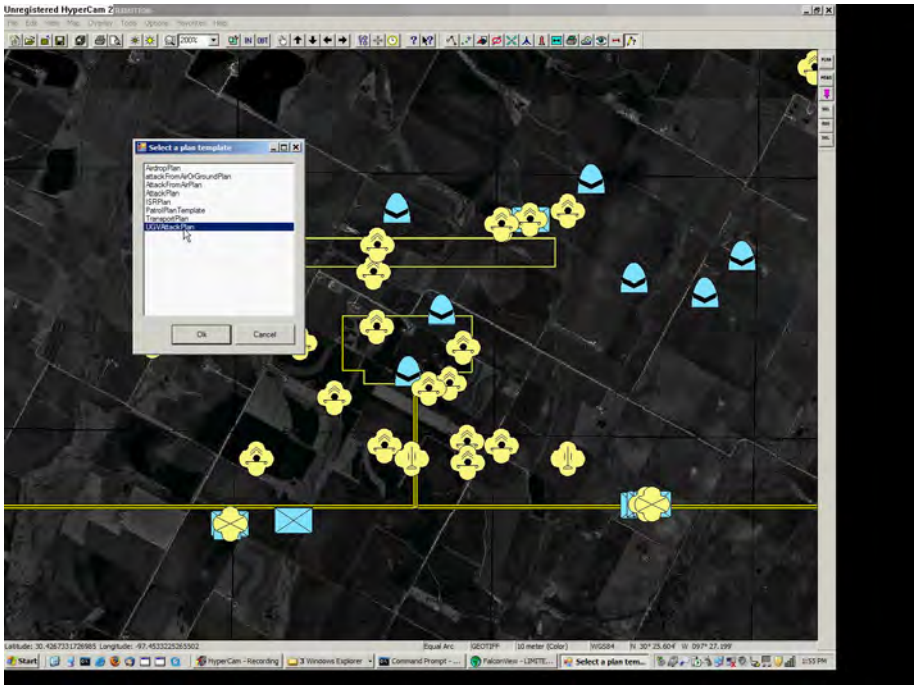

Figure 4. FalconView interface with dialog box for selected a team plan shown.

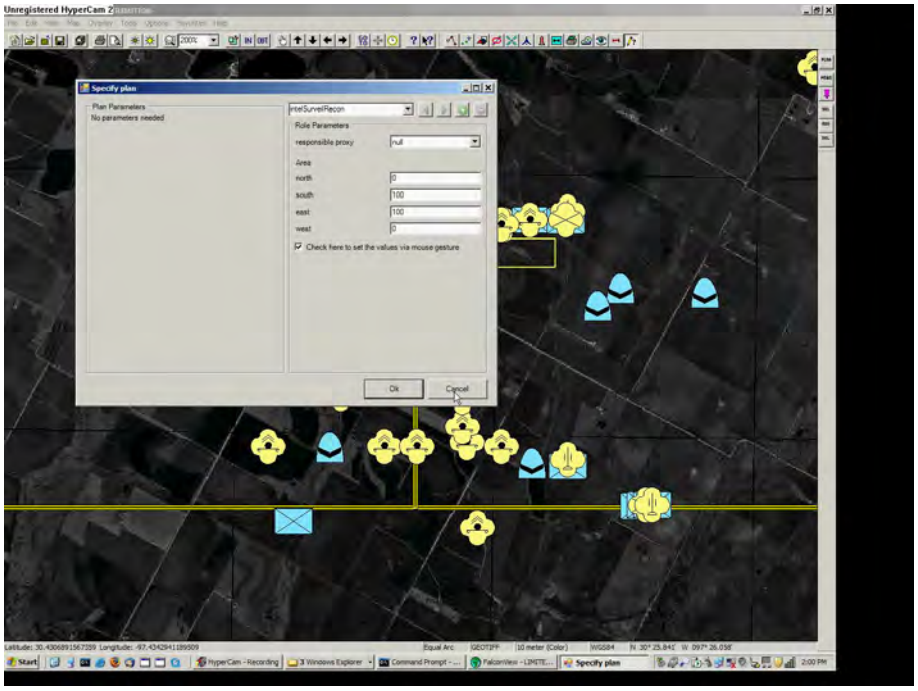

Figure 5. FalconView interface with dialog box for instantiating plan parameters shown.

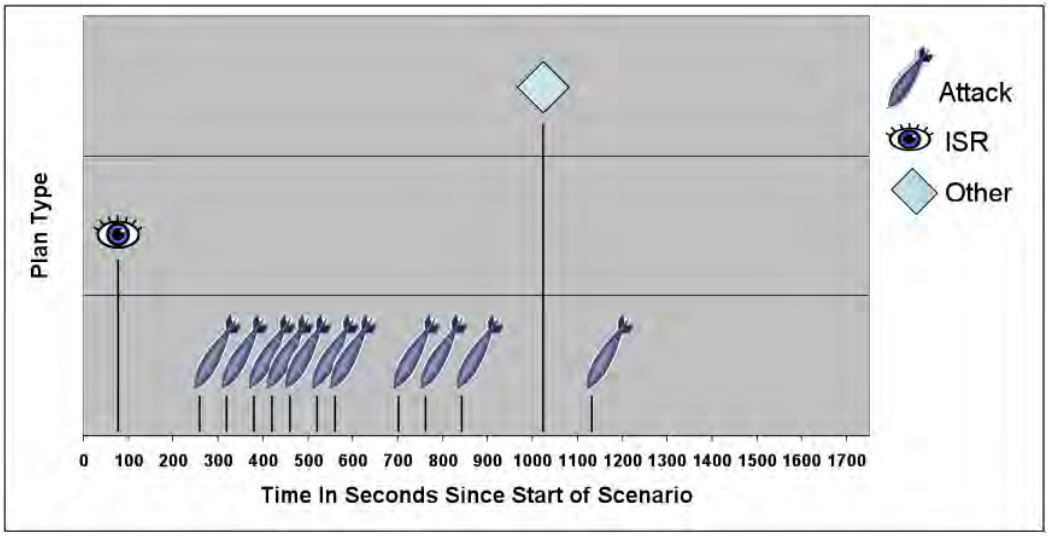

Figure 6. The timeline of team plans initiated by the first subjects.

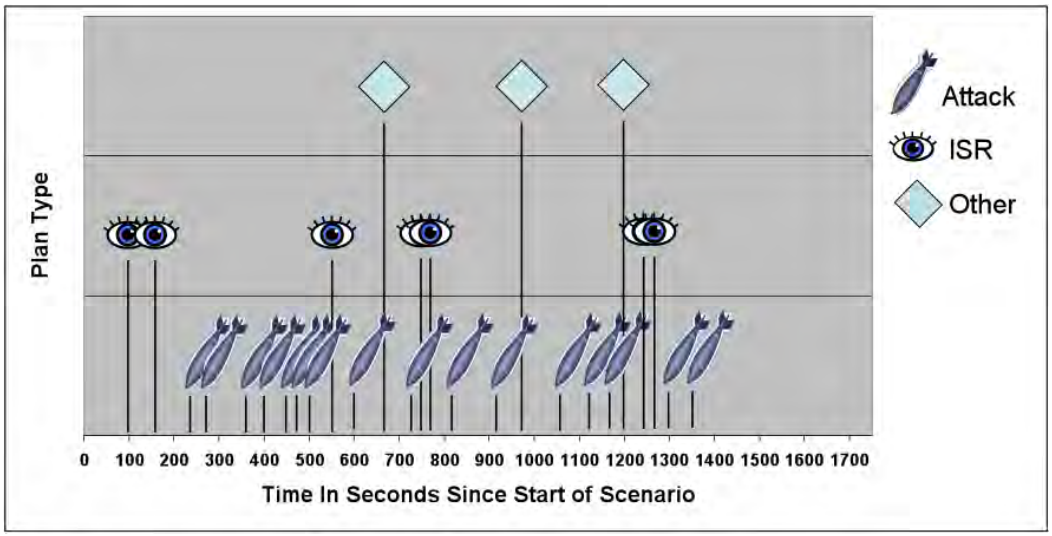

Figure 7. The timeline of team plans initiated by the second subjects.

in. These time lines show the basic flexibility available to the user and how even inexperienced users choose to exercise the power of the team plans in different ways.

Figures 11 and 12 show two additional metrics of performance. Figure 11 shows the number of friendly force casualties incurred by each of the groups. These numbers are reasonably low, given the high degree of firepower available to the opposing forces. The ISR duration gives an indication of how the groups differed in the amount of time they spent performing ISR before sending in humans.

Finally, the table gives the average subjective scores given by the 12 subjects to each of 11 questions. For all questions 5 is highest, 1 is lowest. One surprise in the responses was that the users did not think the team plans were too specific, despite giving them significantly less control than typical unmanned systems GUIs. This was the case even when the users did not feel particularly in control of the situation. This perceived lack of control would need to be addressed before the system could be put to practical use. In informal conversations with users after the testing, one consistent suggestion was to provide more feedback about what the plans are doing. ACCAST currently gives very little information about the plan status.

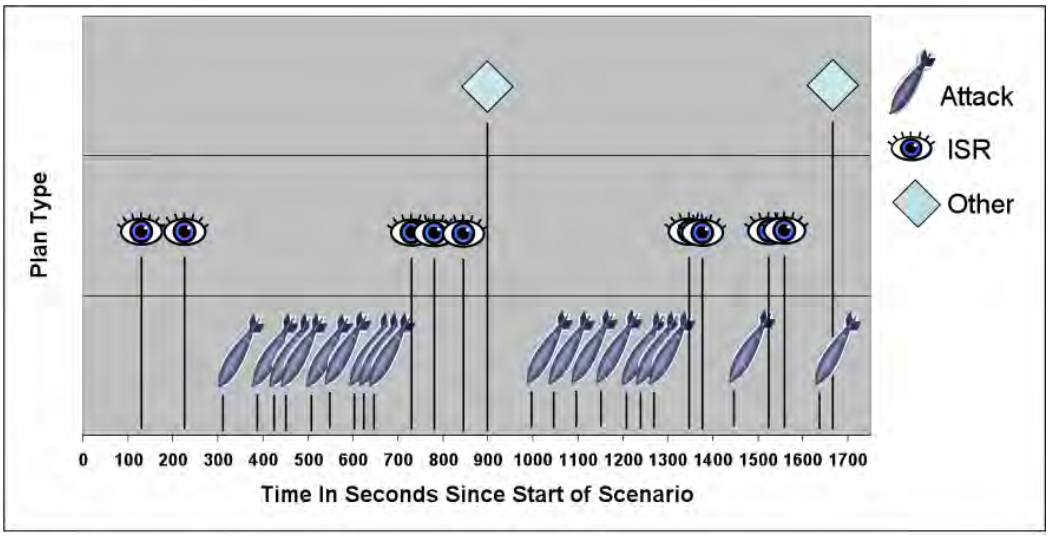

Figure 8. The timeline of team plans initiated by the third subjects.

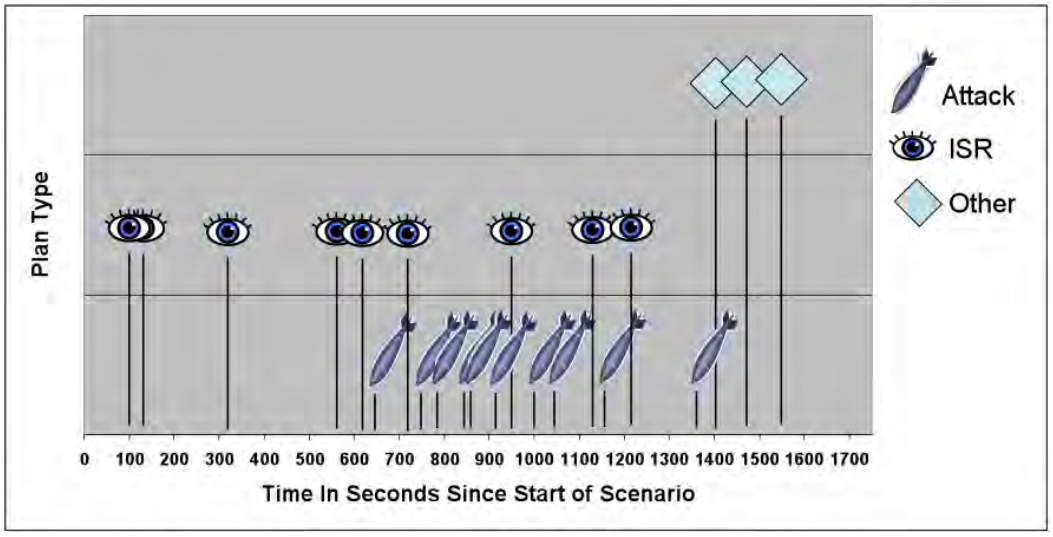

Figure 9. The timeline of team plans initiated by the fourth subjects.

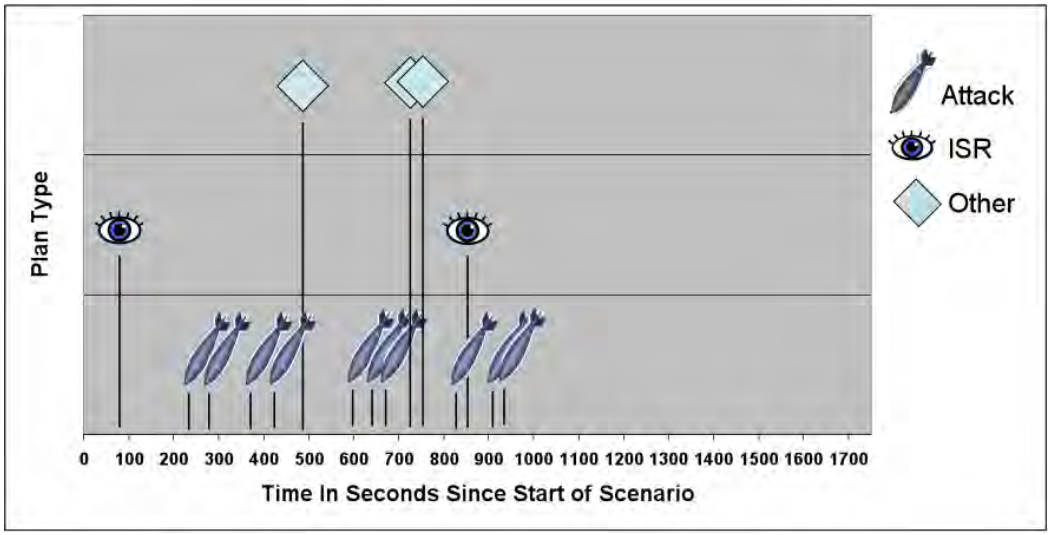

Figure 10. The timeline of team plans initiated by the fifth subjects.

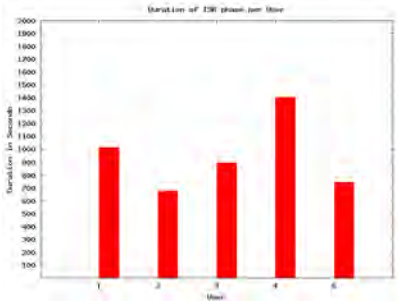

Figure 11. The number of friendly force casualties incurred by each of the groups.

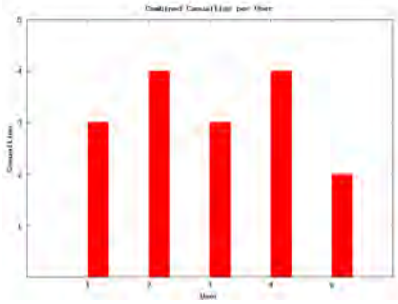

Figure 12. The length of time spent by each group performing ISR before sending in manned vehicles.

Question	Average Score
I felt that I was in control of the situation	3.25
I was able to make the unmanned systems behave as I wanted them to behave	3.5
The available plans were too specific	2.08
The available plans were too general	3.17
I had the flexibility I needed to achieve my mission	3.83
The GUI was intuitive	3.58
The GUI was easy to use	3.75
The GUI felt familiar	3.33
ACCAST would lessen the workload of unmanned systems users	4.08
ACCAST represents an improvement over the current methodology	3.91

6. CONCLUSIONS

The ACCAST project began with the hypothesis that for the military to take full advantage of the large numbers of autonomous vehicles coming into service, it would be necessary to develop techniques that allow a single user to manage many autonomous vehicles. Moreover, it was the ACCAST teams belief that for a single user to manage many vehicles, the autonomous vehicles must be able to autonomously coordinate and that primary mode of interaction must be team lev-el directives that left the details of coordination, path-planning and even task assignment to the team to work out.

Bringing together state-of-the-art technologies from ADI and LM-ATL, the project implemented a system to test this hypothesis. The coordination built on CMUs Machinetta coordination software, communication utilized LM-ATLs FIORO and the interface was developed on top of the FalconView interface. A simulation engine was developed specifically for the project to pro-vide the speed, flexibility and sensing fidelity required of the project. The simulator includes models of many types of assets that might form part of a future FCS company. The simulator allowed investigation of possible conops for working with teams of manned and unmanned systems and is now being used in an Air Force MURI.

Experiments were performed with 21 Army officers at Fort Benning. The results overwhelmingly showed the effectiveness of the system for allowing an operator control a large team. However, issues with feedback and control were identified. Much more extensive experiments using a simulated user, which had the same set of controls as a real user and allowed repeatability and systematic variation, showed the robustness of the control system to a range of factors.

While we believe the project has established the feasibility of a control paradigm using autonomous coordination and team plans as an interaction mechanism, many questions have been raised. Building on top of FalconView limited the range of GUI elements that could be used and the way information and options can be presented. Our initial results show that the operators need a lot more and different information than FalconView is designed to present. We anticipate that dramatically different interface designs are going to be required to present this information. Another interesting point that came up in discussions with subjects at Fort Benning is how this would fit into the overall C2 structure. One possibility would be to break control between a commander with very high level view of the team and an operator that saw much more detail about specific activities of individual vehicles.

ACKNOWLEDGMENTS

The authors would like to thank David Van Brackle and others at Lockheed Martin ATL for their excellent work on this project.

REFERENCES

[1] Miller, C., "Modeling human workload limitations on multiple uav control," in [*Proceedings of the Human Factors and Ergonomics Society 47th Annual Meeting*], (2004).

[2] Crandall, J. W., Goodrich, M. A., Olsen, D. R., and Nielsen, C. W., "Validating human-robot interaction schemes in multitasking environments," *IEEE Transactions on Systems, Man, and Cybernetics, Part A* **35(4):438449** (2005).

[3] Drury, J., Riek, L., and Rackliffe, N., "A decomposition of UAV-related situation awareness," in [*Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*], 88–94, ACM New York, NY, USA (2006).

[4] Cummings, M. and Mitchell, P., "Management of multiple dynamic human supervisory control tasks for UAVs," in [*Human Computer Interaction International Human Systems Integration Conference*], (2005).

[5] Calhoun, G., Draper, M., and Nelson, J., "Advanced Display Concepts for Uav Sensor Operations: Landmark Cues And Picture-In-Picture," in [*Human Factors and Ergonomics Society Annual Meeting Proceedings*], **50**(1), 121–125, Human Factors and Ergonomics Society (2006).

[6] Quigley, M., Goodrich, M., and Beard, R., "Semi-autonomous human-uav interfaces for fixed-wing mini-uavs," in [*Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*], **3** (2004).

[7] Scerri, P., Pynadath, D. V., Johnson, L., Rosenbloom, P., Schurr, N., Si, M., and Tambe, M., "A prototype infrastructure for distributed robot-agent-person teams," in [*The Second International Joint Conference on Autonomous Agents and Multiagent Systems*], (2003).

[8] Pynadath, D., Tambe, M., Chauvat, N., and Cavedon, L., "Toward team-oriented programming," in [*Intelligent Agents VI: Agent Theories, Architectures, and Languages*], 233–247 (1999).

[9] Scerri, P., Farinelli, A., Okamoto, S., and Tambe, M., "Allocating tasks in extreme teams," in [*AAMAS'05*], (2005).

[10] Xu, Y., Scerri, P., Yu, B., Okamoto, S., Lewis, M., and Sycara, K., "An integrated token-based algorithm for scalable coordination," in [*AAMAS'05*], (2005).

[11] Scerri, P., Owens, S., Glinton, R., and Sycara, K., "Synergistic integration of agent technologies for military simulation," in [*KIMAS*], (2007).

[12] McLees, L., "FalconView soars," *AIAA Student Journal* **36**(1), 16 (1998).