Controlling Teams of Uninhabited Air Vehicles

Jeremy W. Baxter and Graham S. Horn
QinetiQ Limited
Malvern Technology Centre
St Andrews Road, Malvern, WR14 3PS, UK
{jwbaxter, ghorn}@QinetiQ.com

ABSTRACT
We describe a Multi-Agent System (MAS) for controlling teams of uninhabited air vehicles (UAVs) in the context of a larger system that has been used to evaluate potential concepts of use and technologies. The approach is one of a decision-making partnership between a human operator and an intelligent uninhabited capability. The MAS controls the UAVs and self-organises to achieve the tasks set by the operator with interaction via a variable autonomy interface. We describe how the agents are integrated with the rest of the system and present a number of system integration issues that have arisen. The overall system has been evaluated in a number of human-in-the-loop trials within a detailed synthetic environment.

Categories and Subject Descriptors
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *multi-agent systems.* I.2.9 [**Artificial Intelligence**]: Robotics – *autonomous vehicles, operator interfaces.* I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, Search – *plan execution, formation and generation, scheduling.* J.7 [**Computers in Other Systems**]: Command and Control, Military, Real time.

General Terms
Design, Experimentation.

Keywords
Multi-agent system, human-machine partnership, variable autonomy, uninhabited air vehicles.

1. INTRODUCTION
Uninhabited vehicles can be used in many applications and domains, particularly in environments that humans cannot enter (e.g. deep sea) or prefer not to enter (e.g. war zones). Uninhabited air vehicles (UAVs) are of particular interest to the defence sector because they have the potential to significantly reduce the risk to aircrew in military operations. The promise of relatively low cost, highly reliable and effective assets that are not subject to the physical, psychological or training constraints of human pilots has led to much research effort across the world. Current systems, such as Predator or Global Hawk, require multiple operators to control a single platform. This paper describes an approach to allow a single operator to control multiple platforms which has been evaluated in a number of human-in-the-loop trials within a synthetic environment (SE).

The concept is one of a decision-making partnership between a human operator and an intelligent uninhabited capability. The human provides mission-level guidance (with support from planning tools, etc.) to the "pool" of co-operating UAVs and takes on a largely supervisory role. The UAVs self-organise to achieve the goals set by the operator. Due to regulatory or liability issues, some critical decisions will have to be made by a human. Therefore, the uninhabited capability must refer such decisions to the operator. This concept has been implemented using a variable autonomy interface onto a multi-agent system, as part of a larger trials system.

The trials system is used to evaluate potential concepts of use and technologies. It is therefore not a static system but one in which different subsystems (such as different human machine interfaces) can be inserted and evaluated. SE based trials enable the key requirements for the decision-making partnership to be captured. The elements of the system have evolved in response to feedback from trials (subjective comments and objective performance measures) and changes to the concepts of use.

A multi-agent system (MAS) provides a natural and powerful way of representing multi-platform tasks and sets of coordinated and cooperating agents. Agents carrying out tasks which are clearly linked to a single platform can be hosted on that platform while more general purpose agents can spread out amongst the platforms. Planning systems can be integrated into the system by producing agents to wrap them.

The remainder of the paper is structured as follows. First we describe the scenario used in the human-in-the-loop trials and how variable autonomy is achieved. An overview of the trials system is given in section 4 before describing the multi-agent system (section 5) and how it is used to control and coordinate the UAVs. Some of the issues we faced when integrating the MAS with the other components are highlighted in section 7, particularly the integration of deliberative planning systems with the reactive planning approach used by the agents. Finally we describe some results from the trials, some related work and conclusions.

2. SCENARIO
The scenario used for the trials was a time-critical targeting mission against a high value, mobile target. A package of four UAVs was deployed to locate and destroy the target. The operator was the pilot of a single-seat fighter, operating outside of the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'05, July 25-29, 2005, Utrecht, Netherlands.
Copyright 2005 ACM 1-59593-150-2/05/0007 ...$5.00.

threat range of ground-based anti-aircraft defences. The UAVs are equipped with (long- and short-range) sensors and weapons. The mission consists of two main phases: search and attack. An example run is described in section 6.

3. VARIABLE AUTONOMY

Variable autonomy is achieved by using the PACT framework [4]. PACT stands for Pilot Authority and Control of Tasks and incorporates a hierarchy of interaction between automated systems and an operator (such as a pilot). At one extreme the operator decides (level 0), at the other the system decides (level 5), and in between are a number of "assisted" levels such as the system providing advice on request (level 1) or asking for permission to carry out a suggested decision (level 3). In the trials we have only used the highest three PACT levels, since the MAS is geared towards task execution rather than the provision of advice.

Key decision points, such as mission phase transitions, that may require operator approval trigger the agents to make PACT requests to the operator, via the Task Interface Manager (TIM). The operator can set the PACT level for each possible PACT request and these values are stored in the TIM. This provides a level of abstraction that means the agents will always send a PACT request and the TIM will determine whether or not to forward it to the pilot. At PACT level 5, the TIM will immediately give approval and then inform the operator that the decision has been made. At PACT level 4 the TIM sends a message to the operator to inform him of the action that the agents intend to take and gives him a short time window to reject the request before the TIM will give approval on his behalf. At PACT level 3 the TIM (and hence the agents) must wait for the operator to approve or reject the request.

4. SYSTEM OVERVIEW

Figure 1 shows the main components of the system that has been used in human-in-the-loop trials. The operator interacts with the system through a human-machine interface (HMI) that allows him/her to task the UAVs and provides situational awareness information about the UAVs and any ground vehicles that have been detected. The Task Interface Manager acts as an interface between the HMI and the agents. It translates button-presses on the HMI into fully populated orders for the agents. It uses information from the agents to drive the situational awareness aspects of the HMI. It also manages the PACT interactions. The agents provide the self-organising system for controlling the UAVs which is central to the implementation of the concept. The agents send commands to controllers on each UAV platform and receive status and sightings information. The platforms exist inside a synthetic environment, which contains platform, sensor and weapon models, and different types of ground vehicles. Integrating all of these components has taken significant effort. Many components have been changed between the trials to investigate different aspects, such as the HMI design or type of weapon.

5. MULTI-AGENT SYSTEM

The agents are structured as in Figure 2. There are 4 types of agent. The User Agent acts as the conduit for tasking from, and information to, the operator. It allocates individual UAVs to tasks. The Group Agents are responsible for planning and coordinating the execution of tasks. Specialist planning agents are used to wrap

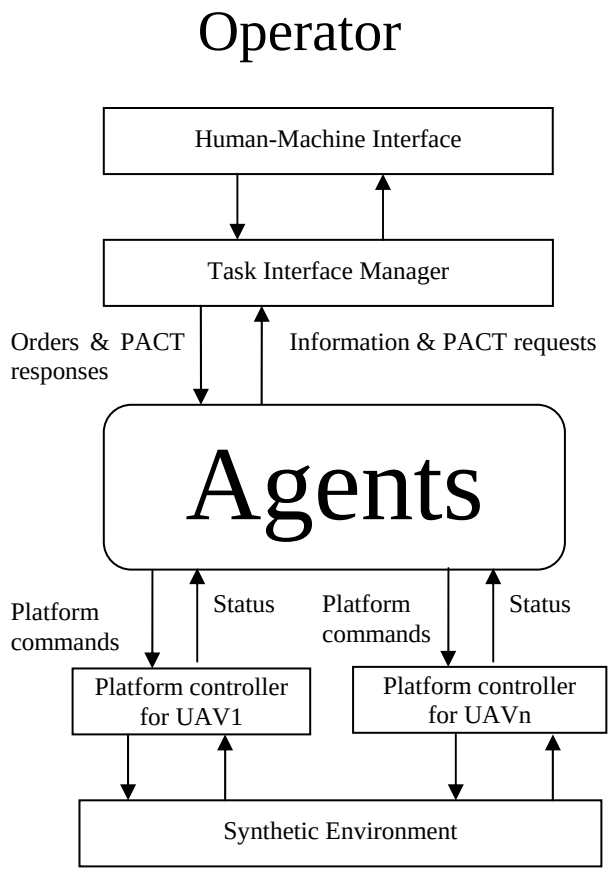

Figure 1. The main components of the system used in human-in-the-loop trials.

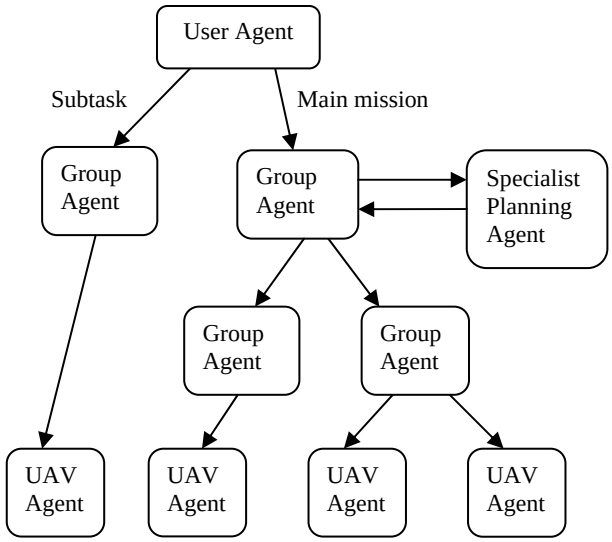

Figure 2. An example command and control hierarchy showing the different agent types.

a number of planning systems to allow them to be used by the agents. UAV agents interact directly with individual platforms, commanding the autopilot to undertake specific manoeuvres.

The MAS was originally designed for controlling teams of entities in ground-based battlefield simulations. In its original form it only contained group agents and vehicle agents and did not allow an operator to issue new tasks during execution (a single order was provided to each team at start-up and this could be decomposed into orders for subgroups). The focus of the original work was on robust execution of these orders in the face of losses and failures [1]. The addition of the user agent allows for operator interaction and parallel tasking.

5.1 User Agent

The User Agent has control of all of the UAVs in the package. It accepts tasks from the operator (via the Task Interface Manager) and attempts to use the available UAVs to complete the tasks. The User Agent has a notion of a 'Main mission' task to which all assets are assigned by default. The User Agent has full control over the task and may change the assignment of any of the UAVs involved in it. In addition to this default task, the operator may specify a number of 'subtasks' which the User Agent is also responsible for trying to achieve. A subtask represents some specific action which the operator may require to be undertaken in addition to the default task, such as observing a specified location. The User Agent must select the assets to carry out each subtask. (In some trials the operator was able to specify which assets to use, if he so desired.) Once the User Agent has identified the UAV assets required for a task these assets are assigned to a Group Agent. Group Agents can be created by the User Agent or existing Group Agents can be re-used. The User Agent therefore controls a number of concurrent tasks on behalf of the user and uses the Group Agents to plan and supervise these tasks.

5.2 Group Agents

Group Agents exist to control a team of UAVs for a single task. Group Agents may either control UAV Agents directly or may control other Group Agents. For example, if a task requires the UAVs to operate in two pairs a Group Agent will control the group of four by tasking two Group Agents (each of these will control a pair of UAVs). Group Agents embody the knowledge of how to plan and execute coordinated team tasks using a framework (described in [1]) based on Joint Intentions theory [5]. This provides a solid grounding for the required communication necessary to keep a team task coordinated. Given an assigned task and assets a Group Agent makes a plan to achieve the task. It may call on additional specialist planning agents to do this. The plan is structured so that the roles which need to be fulfilled are clearly identified and UAVs are assigned to these roles. The plans include the coordination necessary to execute the plan. These plans then form the tasks for subordinate Group or UAV Agents and are sent to them for further planning and execution.

The primary group behaviours are:

- Search for a target, using short-range imaging sensors.
- Attack a target, combining weapon delivery and battle damage intelligence gathering.
- Search & Destroy, which combines the above two behaviours.
- Fly a route in formation.
- Standoff search using long-range sensors.

5.3 Specialist Planning Agents

Two such agents have been used in our system.

5.3.1 Search Agent

The Search Agent provides access to a planner that produces search routes. The planner is provided with a set of possible target positions and expands them into regions that could be reached by a moving target in the next few minutes. The search routes are designed to allow the UAVs to search these regions with short-range sensors and take images of potential targets that will be classified by the operator.

5.3.2 Attack Agent

The Attack Agent provides access to a dynamic scheduler that allocates UAVs to the tasks that must be carried out during the attack phase: release the weapon and gather images of the target after the weapon has detonated to see if it has been destroyed. Typically two UAVs are available for these tasks, which may be split between them or one UAV may be chosen to undertake all tasks. The aim of the scheduler is to minimize the time taken to hit the target and get visual confirmation of its destruction.

The dynamic scheduler implements a deliberative planning process, derived from sequential decision theory, but specialised to *weakly coupled* systems (in which execution of tasks is decoupled after resource assignment) and with appropriate task models could be used to plan a wide range of behaviours. The scheduler implements joint planning up to some time horizon, beyond which uncertainty in the scenario is expected to invalidate attempts to form longer-term plans. In scoring proposed plans, it

makes use of *task models* to evaluate the effect (in terms of state, time and cost) of assigning particular resources (UAVs) to particular tasks. These task models can be stochastic (allowing for uncertain outcomes), but in this application only deterministic task models were used. The dynamic scheduling technology provides an upgrade path in which longer-term (tactical and strategic) considerations can be taking into account through the use of a *value function* that is evaluated at the planning horizon and added to the score of each plan. The value function can be hand-designed or acquired by trial-and-error learning in simulation (reinforcement learning).

5.4 UAV Agents

A UAV Agent exists for each UAV platform. It sends commands to the sensors, weapons and autopilot via a lower level platform controller. The UAV Agent monitors the status of the vehicle and sends sensor and state information to the other agents. The UAV Agents can plan and execute single vehicle tasks, such as taking images of a specified ground entity, and can try different actions until they achieve the tasks set by the Group Agent.

The primary vehicle behaviours are:

- Fly a specified route (possibly taking images of potential targets).
- Take an image(s) of a ground vehicle (fly into position to do so, if necessary).
- Loiter observing a ground vehicle using an imaging sensor.
- Search an area using long-range sensors.
- Release a weapon.

6. EXAMPLE RUN

This section describes a typical run of the system, highlighting the actions taken by the agents and the interactions they have with the operator. At the start of the run four UAVs are on a pre-planned ingress route heading for the last known location of the target. As they close the operator orders that a stand off search of the area should be conducted, using long-range sensors, to build up a picture of activity within the search area. The agents interpret this as a subtask from the main mission. The User Agent selects the two UAVs on the outer edges of the formation to perform this stand off search. Figure 3 shows how the agents have re-organised to accommodate this new structure. The User Agent is controlling two Group Agents, one representing the pair carrying out the stand off search and the other representing the pair continuing the ingress. The operator then specifies the main mission to be 'Search & Destroy' and the User Agent automatically selects the two remaining UAVs carrying out the ingress to perform this task. The Group Agent assigned to the 'Search & Destroy' task identifies that since no target has been classified it should enter the 'Search' phase and try to find one. This Group Agent initiates a Search Agent, which calculates appropriate search routes which are then sent to the UAV Agents to execute. These agents fly their assigned routes and gather images of potential targets that have been identified and tracked by the stand off assets. The agent organization for this phase is shown in Figure 4.

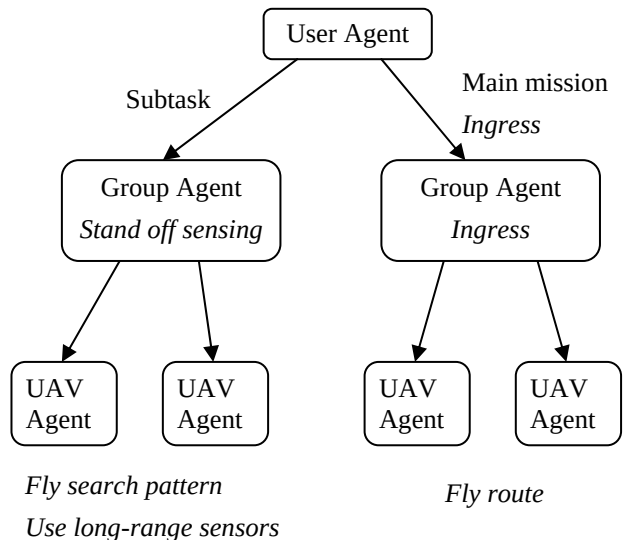

Figure 3. Agent organization for ingress and stand off sensing tasks.

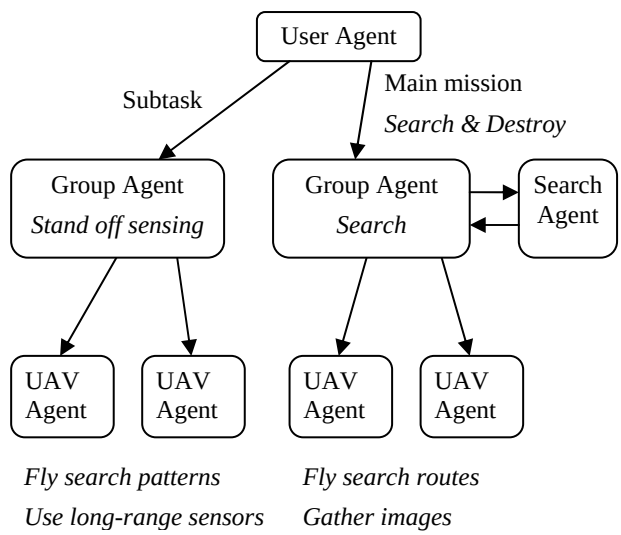

Figure 4. Agent organization during the search phase.

When the operator has classified a ground entity as the high value target the agents infer that the search phase must be over and issue a PACT request, asking for permission to end the search and enter the attack phase. In this example we will assume the PACT level for this decision was set to be five, so the user is presented with a message informing him that the search is over and that the package is entering the attack phase. The Group Agent now uses an Attack Agent to produce a plan, as shown in Figure 5. In this example, one UAV is tasked to release the weapon and the other is tasked to gather battle damage intelligence (BDI) images after the weapon has impacted. When the first UAV is in position to release the weapon it issues a PACT request seeking final launch authorisation. This critical decision point has been set to level 3 and so the UAV cannot release the weapon until the user accepts the request. A timer is displayed on the HMI to indicate the

launch window – if the operator does not respond before the launch window closes then the request will be withdrawn and the UAV will reposition itself before asking again. Assuming that the operator accepts the initial request promptly, the weapon is released. The BDI images are sent to the operator who confirms the destruction of the target.

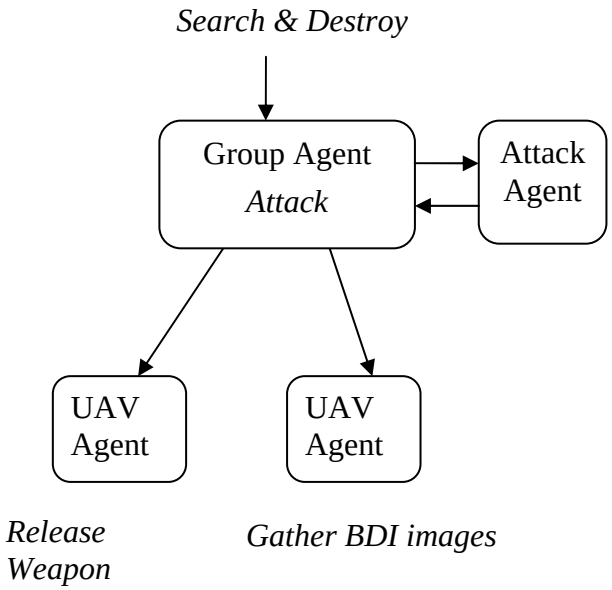

Figure 5. Agent organization for the attack phase of the search & destroy mission.

7. SYSTEM INTEGRATION ISSUES

Assembling the components for each trial has required significant effort. Compromises have had to be made to make things work given tight time constraints. This section highlights some of the issues faced when integrating the agents with other subsystems.

7.1 Reactive and Deliberative Planning

The agents are implemented using a Beliefs-Desires-Intentions agent language. We have built our own coordination framework on top of this to provide robust execution of group tasks. Group agents and UAV agents have sets of reactive plans to carry out group and individual tasks. Deliberative planning is done by the specialist planning agents. The search planner and the attack scheduler both contain models that are used to produce plans based on predicted future world states. Mixing deliberative and reactive planning can lead to problems. In particular, the task sequences produced by the scheduler are executed by UAV agents using reactive behaviours which differ from the models used for planning. This means that tasks may be completed in more (or less) time than expected and this may have a negative impact on the rest of the plan. Making use of the *stochastic* task models in the scheduler would provide better robustness to uncertainty in the outcome of actions. Closer integration of the deliberate and reactive elements would be helpful, such as sharing data structures (e.g. tasks, constraints, and plans).

Another issue is re-planning authority. The scheduler generates new plans on a regular basis, and when it finds one that is better than the current plan by some threshold the new plan is sent to the group agent in charge of the attack task. Currently, the scheduler does not take sufficient account of the cost of interruption and this can lead to delays rather than reducing the time to prosecute the attack.

The task sequences produced by the attack scheduler do not contain any coordination information (the tasks may contain preconditions based on time). Due to the number of different assignments that the scheduler might make (potentially all four UAVs might be available), we did not make full use of our coordination framework and fell into the trap of using an *ad hoc* coordination mechanism which was not sufficiently robust. Re-implementation to make use of our coordination framework was not possible in the time available before the trial which used the attack scheduler.

7.2 Communication

The trials system consists of a number of components (Figure 1). These components have been developed by different groups in different programming languages. Interaction between the agents and other components, namely the TIM and platform controllers, is via XML messages over sockets. Whilst this is a verbose method of communication it has several important advantages over other message formats. XML messages are human readable, and this greatly assists debugging. It allows components to be easily replaced. The HMI and TIM used for the trials have been replaced with a ground control station without impacting the other parts of the system. It is also easy to add new message types or to modify the message specifications. An earlier version used Java interfaces to separate the agents from the platform controllers, with the platform controller team providing implementations that would work with their component.

7.3 Logging and Debugging

In large distributed systems it is important to be able to understand what is happening (both during development and for post trials analysis) and to trace faults. Since the trials system is highly distributed each subsystem produces separate log files. The main log file generated by the MAS is an XML document which records all messages sent between the agents and 'debug output' which provides information on which behaviours are being executed and values that they are using. This log shows how beliefs get propagated around the MAS. It is very difficult to maintain mutual belief and an important issue for future work is to look more closely at reasoning about when to send information to other agents.

We have written a tool to allow browsing of these files, and filtering of the information displayed according to the timestamp, message sender / receiver, message type, or 'debug output' key / agent. The use of this tool has reduced the amount of time required to produce a summary of a run (including investigating potential faults) compared to using a text editor. An obvious issue is the accuracy of the timestamps. Every agent in the system has its own clock, which is set to a common time (based on the simulation time from a randomly selected UAV platform, accessed via the platform controller) at start-up. The initialisation of the agents does not take zero time, so these clocks will have some variation (less than a second, typically). This is an issue for any distributed system.

An example of the use of the logs for tracing faults is a problem that we had where the pilot was not receiving images that we thought should have been taken by the UAVs. Analysis of the MAS log showed that the UAV agents were not receiving notifications that images had been taken and in fact requests to take the images had not been sent to the platform controller, so the appropriate behaviour was modified to put the UAV into a better position in order to take an image of a ground vehicle. This fixed that part of the problem. Further analysis showed that the requests were now being sent to the platform controller but it was not sending them to the simulation because it had different values for the sensor field of view from those used by the agents and the simulation. When this was corrected we found that the images were taken, but sometimes the pilot did not receive them – due to an unreliable transport mechanism within the SE (an unexpected problem that thankfully did not occur often). Without our logging and support tools it would have taken much longer to identify and rectify these problems and they quickly justified the time and effort spent developing them.

7.4 Test Harness

Running the entire trials system requires a large number of resources (computers, people and time). In order to develop the agents separately we have produced a test harness which consists of a (lower fidelity) simulation and a number of displays (to replace the HMI and TIM). The test harness and the agents can be run on a single PC. The platform controllers and dynamics models are embedded within our simulation. The test harness has minimal start-up time, allowing a greater proportion of time to be spent testing behaviours and integrating the specialist planning systems. However, it is important to recognise the risk that behaviours may be prematurely optimised with respect to the test harness rather than the detailed synthetic environment.

8. TRIALS RESULTS

8.1 System Performance

The multi-agent system has now been used in three successful human-in-the-loop trials. Results have shown that the approach is a good match to the concept of a human – intelligent system partnership. The pilots were able to successfully control a team of four assets to complete the missions. One of the problems in the first two trials was that the time taken to carry out the attack phase was too long, and this is why a separate attack scheduler was developed before the third trial. Fielding the system within a test and evaluation environment has revealed some implicit assumptions that the developers of different subsystems have made.

The PACT framework worked well when the pilots and agents were in agreement on the situation and the appropriate response. When the pilot refused a PACT request it was often unclear what the appropriate response of the system should be. For example, in some situations it might be appropriate to suggest an alternative, in others it might be appropriate to wait a short period of time and re-issue the same request. Usually the refusal of a PACT request indicates the there is a mismatch between the agents' assessment of the situation and the pilot's. In the current system it is usually the case that a re-tasking by the operator has to follow a refused PACT request but future work plans to look at alternative ways of dealing with this problem.

8.2 Trust

One of the aims of this work is to reduce operator workload, so that a single operator can comfortably control a pool consisting of multiple UAVs, and one of the purposes of the trials is to investigate how much information is needed by the operator. A key requirement is a shared awareness between the operator and the intelligent capability about the current situation and what to do about it. The HMI must therefore enable the operator to understand what the agents believe the current situation to be and to provide the agents with the operator's view of the situation. The former is achieved through status reports that indicate the high level mode and objectives of the agents (which allow the operator to gain an understanding of the progress – at a suitable level of abstraction – of the tasks he has issued) and messages from the agents. Messages may be informative or alerts. Determining how many messages (and their priority) and the level of detail to show to the operator is difficult, because a terse system may be harder to trust, whereas a verbose system will overload the operator. Alerts can signal execution failures that the agents are unable to resolve without user assistance. The most recent trial has shown that these failures need to be communicated more clearly.

9. RELATED WORK

The variable autonomy provided by PACT can be considered to be user-based adjustable autonomy as described by Maheswaran et al [6]. The key decision points at which PACT requests are issued by the agents cover the two classes of policies in their framework, where a weapon launch request is an example of *permission requirements for action execution* and a request to advance the mission phase is an example of *consultation requirements for decision making.* Agent-based adjustable autonomy, where the agent decides when to transfer decision-making control to another entity, is not supported in our system.

Hierarchical architectures are an obvious choice for controlling uninhabited vehicles. Three examples follow.

Howard et al [3] present a three-layer model where the lowest layer (the action layer) is equivalent to the platform controllers in our architecture. Their single-agent layer is equivalent to our UAV agents. UAV agents, group agents and the user agent have aspects of their teamwork layer. Their hierarchical processing is instantiated by additional teamwork layer processes on some of the UAVs. These additional processes fill the role of the Group Agents and User Agent in our system. A bidding protocol is used to allocate tasks to UAVs or subgroups.

Chandler et al [2] present a hierarchical agent architecture for controlling multiple vehicles. At the top is an inter-team cooperative planning agent which is equivalent to our User Agent. It uses an auction procedure to allocate observation targets to teams of UAVs. Below this are intra-team cooperative control planning agents (equivalent to group agents) which send tasks to vehicle planning agents (UAV Agents). At the bottom are UAV regulating agents which provide command sequences for the vehicle, control sensors, etc. (functionality provided in our system by UAV agents and platform controllers).

Vachtsevanos et al [9] present a generic hierarchical multi-agent system architecture with three levels. Agents in the upper level mainly provide decision support tools for the mission commander,

with a focus on global knowledge for producing team plans. Our group agents and specialist planning agents provide many of these functions. The middle level is responsible for planning and monitoring the tasks of a single UAV and is equivalent to our UAV agents. The lower level consists of a set of agents that control the vehicle, sensors, weapons, etc and is designed to support heterogeneous UAV models. This functionality is provided in our system by the platform controllers (with some overlap with the UAV agents).

Our coordination framework (described in [1]) bears a close resemblance to the STEAM rules [7] (and the subsequent TEAMCORE work [8]) produced by Tambe et al, which is also based on Joint Intentions theory. The main difference is the presence of an agent representing the group as a whole that is responsible for instructing and coordinating the group members, as opposed to team members simultaneously selecting joint operators.

10. CONCLUSIONS

The approach described in this paper has been evaluated in a number of human-in-the-loop trials within a synthetic environment and it seems to be a good match for the concept of a decision-making partnership between a human operator and an intelligent uninhabited capability. The overall trials system provides a framework for evaluating concepts of use of potential technologies.

The multi-agent system is able to self-organise to achieve the tasks set by the operator. The PACT framework for variable autonomy worked well when the operator and agents were in agreement, but further work is needed to cope with cases were the operator rejects PACT requests.

11. ACKNOWLEDGMENTS

This work was funded by the research programme of the United Kingdom Ministry of Defence under contract DTA 3e 201.

12. REFERENCES

[1] Baxter, J. W. and Horn, G. S. Executing Group Tasks Despite Losses and Failures. In *Proceedings of the Tenth Conference on Computer Generated Forces and Behavioral Representation*. Norfolk, VA, 15-17 May 2001, 205-214.

[2] Chandler, P. R., Pachter, M., Nygard, K. E. and Swaroop, D. Cooperative control for target classification. In *Cooperative Control and Optimization*, edited by Murphey, R. and Pardos, P. M., Kluwer Academic Publishers, May 2002.

[3] Howard, M., Hoff, B. and Lee, C. Hierarchical Command and Control for Multi-Agent Teamwork. In *Proceedings of the 5th International Conference on Practical Applications of Intelligent Agents and Multi-Agent Technology (PAAM2000)* Manchester, UK, April 2000.

[4] Howitt, S.L. and Richards, D. The Human Machine Interface for Airborne Control of UAVs. *2nd AIAA Unmanned Systems, Technologies, and Operations Aerospace, Land, and Sea Conference and Workshop*. September 2003.

[5] Levesque, H., Cohen, P. and Nunes, J. On Acting Together. In *Proceedings of the Eighth National Conference on Artificial Intelligence* (AAAI-90) (Boston, MA). AAAI, Menlo Park, CA, 1990, 94-99.

[6] Maheswaran, R. T., Tambe, M., Varakantham, P. and Myers, K. Adjustable Autonomy Challenges in Personal Assistant Agents: A Position Paper. In *Agents and Computational Autonomy: Potential, Risks and Solutions*, edited by M. Nickles, G. Weiss and M. Rovatsos, Springer-Verlag, 2004.

[7] Tambe, M. and Zhang, W. Towards flexible teamwork in persistent teams. In *Proceedings of the International conference on multi-agent systems (ICMAS)*, 1998.

[8] Tambe, M., Shen, W., Mataric, M., Goldberg, D., Modi, J., Qiu, Z., and Salemi, B.: Teamwork in cyberspace: Using TEAMCORE to make agents team-ready. In the *Proceedings of AAAI Spring Symposium on Agents in Cyberspace*, 1999.

[9] Vachtsevanos, G., Tang, L. and Reimann, J. An Intelligent Approach to Coordinated Control of Multiple Unmanned Aerial Vehicles. *Presented at the American Helicopter Society 60th Annual Forum*, Baltimore, MD, June 2004.