Experiences with the Design and Implementation of an Agent-based Autonomous UAV Controller

Samin Karim*
Department of Information Systems
University of Melbourne
Melbourne, Australia
mkarim@pgrad.unimelb.edu.au

Clint Heinze
Air Operations Division
Defence Science and Technology Organisation
Melbourne, Australia
clinton.heinze@dsto.defence.gov.au

ABSTRACT

This paper reports experiences and outcomes of designing and developing an agent–based, autonomous mission control system for an unmanned aerial vehicle (UAV). Most UAVs are not truly autonomous or even unmanned but are more correctly termed 'uninhabited' or 'remotely piloted'. This paper explores two quite different approaches for adding autonomous control to an existing UAV. Both designs were implemented using an agent–based language. The first takes a fairly standard approach, adding a layer over the flight control system to control the mission. The second takes the human metaphor of agency more seriously and implements an autonomous controller based on a model of human decision making widely referenced in the military command and control literature. Implementing these two designs allowed a comparison of their relative strengths and weaknesses. Preliminary findings indicate both the feasibility and usefulness of a human cognitive modelling approach to providing autonomous UAV control but indicate a number of important considerations. This paper also reports on the successful first flight trials of the Codarra Avatar UAV under the mission control of the agent and discusses the future flight test program.

Categories and Subject Descriptors

D.2.10 [**Software**]: Software Engineering—*Design*
; I.2.9 [**Artificial Intelligence**]: Robotics—*Autonomous Vehicles, Commercial robots and applications*

General Terms

Design, Experimentation, Languages, Theory

Keywords

UAV, real-time agent-based control, cognitive robotics, cognitively plausible models, OODA, BDI agency

1. INTRODUCTION

The proliferation of unmanned aerial vehicles (UAV) for civilian and military applications has fostered research in many areas. The application of intelligent agent technologies in this domain is a natural one, if embryonic at present, but most agent research in this area has focussed on swarming technologies and means of generating emergent yet controllable intelligent team behaviour or as subsystem managers [16, 19, 9, 15]. Indeed most UAVs are neither autonomous nor unmanned but are more correctly regarded as *remotely piloted* or *uninhabited* and many of the larger UAVs require significant human control by telepresence. This is particularly true of the control of teams of collaborating UAVs, which often require significant human intervention during coordination tasks.

Attempts to build autonomous UAVs often adopt a design approach that augments a fairly standard autopilot with mission control layers. This is unsurprising, given the conservative safety focussed nature of the aviation industry. This approach is not inconsistent with Brooks' subsumption architecture [5] where the generation of intelligent behaviours is undertaken with minimalist control systems that provide tailored behaviours without the need for sophisticated representations of the environment. This paper explores a different direction by testing the hypothesis:

> *The development of an autonomous controller for a UAV is assisted by the adoption of human cognitive models as the basis for the software architecture, and that this notion extends to the decentralised coordination of multiple UAVs*

With regard to decentralised (multiagent) coordination of multiple UAVs, the concept of using human cognitive models to design the mission management systems of UAVs is an interesting one. For instance, well established communication and coordination protocols employed by pilots, in military and non-military domains, can be equally applied to autonomous UAV control systems. Furthermore, by 'getting inside the head' of human controllers (such as pilots), in common control and coordination scenarios, some interesting insights can be ascertained about the way autonomous agents within *real-time*, multiagent scenarios should behave and act. The experiments presented in this paper do not tackle the issue of UAV or, indeed, multiagent coordination.

*Primary author is a PhD research student

AAMAS'05, July 25-29, 2005, Utrecht, Netherlands.
Copyright 2005 ACM 1-59593-150-2/05/0007 ...$5.00.

However, the idea of using cognitive models may naturally extend to coordinated *teams* [22] (see §6 for proposed future work in UAV coordination).

In simulation and other virtual environments where the constraints of real-time are often absent and risks are relatively low it has been demonstrated that adopting human cognitive models as the basis for designing entity controllers can deliver knowledge engineering and software engineering advantages. In particular, agent technologies have provided the basis for the construction of models of military pilots [11].

This paper will compare these approaches by presenting two alternative mission controllers (§4). Both make use of the JACK Intelligent Agents platform for their implementation but employ quite different designs. The first adds a simple mission-control layer matched to the flight control system. The second adheres more strongly to the *agent metaphor* and implements a design based on Boyd's observe-orient-decide-act (OODA) loop model of military decision making [4]. In §3 the first flight trials of the UAV under the autonomous mission control of the agent are reported and in §6 the future flight trials program is outlined.

2. BACKGROUND

This section briefly references the important technologies that were brought together during this development and outlines the ongoing science and technology program of which this paper is a part.

2.1 Agent-based real-time controllers

The agent-programming language, JACK[1], has been used for the design and implementation of the mission control software, which adds intelligent, autonomous decision making to the standard autopilot Flight Control System (FCS) of the test platform, the Codarra Avatar UAV (refer to §2.3 and §4.2 for more details). JACK is a programming language with the essential programming construct called *agents*. Agents are abstracted entities akin to objects in the object-oriented paradigm. However, agents in JACK are autonomous entities that exhibits intelligent behaviours structured around the BDI, (**B**elief **D**esire **I**ntention) theory of agency [21].

The BDI framework supports a goal-directed reasoning process. Each agent pursues its given goals (**desires**), adopting and committing to appropriate plans (**intentions**) according to its current set of data (**beliefs**) about the state of the world. In this way, agents behave as folk-psychologically plausible caricatures of humans with the mental attributes encompassed by the BDI framework.

The combination of desires, goals and context-sensitive intended behaviour is the fundamental characteristic of BDI agency. There are several benefits in using agents and specifically the BDI framework of agency over other frameworks:

- **Modular Design.** The system can be decomposed and organised into functional components that are autonomous and execute in parallel. Each intelligent, autonomous component (agent) serves one or more purposes, and the overall system function emerges from the interaction of all agents that form the system. Furthermore, *capabilities* provide JACK with structure and modularity.
- **Abstraction.** The high level of abstraction that agents provide allows the designer to construct a system based on an *organisational unit*, or *team*, that exhibit human-like behaviours and cognitive mechanisms. This is an intuitive and familiar structure that facilitates both the design and the implementation process. Further to the previous point, *capabilities* also provide JACK with an additional level of abstraction.
- **Scalability.** *Multiagent systems* is a burgeoning research area with several frameworks under development and in commercial use (such as STEAM [23], TEAMCORE [24], Soar [18] and JACK Teams [2]). Reasoning, communication and coordination protocols are explicit components of these systems. One common theme with the more developed frameworks, such as STEAM and JACK Teams, is that coordination is handled in a flexible, domain independent manner, as opposed to implicitly 'hard-coding' domain-specific coordination plans. For a team-oriented, dynamic, real-time application such as coordinating UAV teams, the arguments for employing a multiagent approach to team coordination are compelling.

2.2 Simulating Pilots and the OODA Loop

In conflict based computer games and in military simulation it is necessary to simulate the behaviour of pilots (and other human participants) in virtual battle-spaces. In these domains it is sometimes the case that the behaviours required of the virtual entities not only appear plausible but in a very real sense *be* plausible. They should respond to situations and commands, to interactions with real humans and to provide some capacity for their decision making processes to be inspected, explained and validated. In these situations it has proven useful to construct and maintain models of decision making that accord with the folk ascriptions of their behaviour provided by subject matter experts.

Experiences with the development of simulations of pilot decision making have resulted in the adoption of the BDI model as the basis for providing the programming-level building blocks necessary for that folk-psychological familiarity. Furthermore Boyd's OODA loop model of military decision making is widely known by pilots and is often used by them, sometimes unconsciously, to describe their decision-making processes [14]. The OODA loop model of decision making has been influential over the last decade in both military and business circles [20]. Whether or not it has value as a model of human decision-making it has certainly provided a useful architecture for structuring the development of intelligent agents [12].

Whether or not experiences developing virtual pilots for simulators translate to the design of agents for the autonomous control of real aircraft is one of the open questions considered by the research program that led to this paper.

2.3 The Codarra Avatar UAV

Before discussing the issues in giving a UAV autonomous abilities, it is worthwhile to briefly discuss the engineering and system attributes of our particular UAV platform, in addition to considering the agent architecture.

The *Codarra Avatar* [1] is a lightweight UAV, purpose-built for small-scale reconnaissance and surveillance missions. Codarra initially perceived a need for tactical surveillance for military platoon-sized patrols. A small, programmable

[1] http://www.agent-software.com

UAV was developed to address this need. The Avatar can be rapidly disassembled and reassembled, launched by hand (ie. thrown in the air for take-off), and easily transportable in a back-pack. The propulsion system consists of a lithium-polymer battery and a small electric motor driving a fully-folding propeller. Flight endurance is approximately 60 minutes. Recovery (landing) is by a parachute that deploys on command from a cavity above the fuselage centre-section. Payloads vary from a video camera capable of transmitting 6 MHz video bandwidth, to any other sensor up to 1.5 kg in weight and about the size of a standard house brick: this includes the iPAQ personal digital assistant (the platform for the agent-based mission controller) carried for the first set of trials. The Avatar has on-board GPS for basic navigation, an airspeed indicator and barometric altimeter.

Applications of the Avatar are diverse: mobile tactical reconnaissance, surveillance, intelligence gathering, law enforcement (such as policing and security operations), search and rescue, land management, environmental monitoring, disaster management, and stock and station control. The relatively small size makes it cost effective, energy efficient, easily transportable, flexible and scalable (ie. multiple UAVs can be readily deployed for situations where a multiagent approach is more suitable [15]). Some pictures of the Avatar can be found in §3, Figure 2.

2.4 Issues in Providing UAVs with Autonomy

Significant issues in providing UAVs, and specifically the Codarra Avatar UAV, with autonomy are as follows:

Flight time/range As the UAV is battery powered, the relative flight time/range is small compared to, for example, a standard petrol powered radio-controlled (RC) miniature aircraft. For missions requiring longer flights, this can be achieved by 'refuelling' stops (ie. the UAV returns to base for a battery change, or another UAV 'baton changing' for re-launch).

Durability The Avatar is a lightweight, small aircraft, and therefore it cannot withstand turbulent, high speed wind conditions. In certain conditions, the Avatar can overcome these situations by carefully planning flight paths. Such planning can be executed by the JACK agent mission manager onboard the UAV. However, there are situations where no amount of careful mission planning, by either the JACK agent or a human operator, can overcome the adverse condition.

Flight regulations and restrictions Federal law requires that all aircraft pass through stringent safety checks. In particular, software must be validated and demonstrated to be reliable and safe within conservative margins. Software that exhibits emergent behaviours creates difficulties for manufacturers wishing to develop UAVs based on these technologies - an area currently under consideration by Lockheed Martin Aeronautical Systems' Verification and Validation of Intelligent and Adaptive Control Systems (VVIACS) project. This means that the aircraft cannot be overly 'creative' and behave in a potentially random manner in certain known, and indeed, unknown situations. The JACK programming environment, and specifically the BDI framework of agency, that is being used as the MM programming platform, is ideal in this respect, as the BDI plans that it executes are known at compile time and are executed in a deterministic manner.

Limited computational power As the payload of the Avatar is relatively small, the size of the agent-based Mission Management (MM) processing computer must also be small and subject to limitations in computational power. In our initial experiment, we used the *HP iPAQ* - a standard Pocket PC-based PDA (Personal Digital Assistant), which was limited both in function and in computational power. The iPAQ will most certainly be replaced by the *PC104* or similar cut-down PC platform in future test flights. A more thorough discussion follows in §4.2.

Limited Sensory Data As described at the start of this section, the Avatar is limited to GPS, gyroscopes and wind speed data, with the exception of other data that it may receive from the Ground Control Station (GCS). However, for longer range missions where conditions within the vicinity of the GCS are different from that of the Avatar, this is not a practical arrangement. For such complex, long-range missions, a more sophisticated suite of sensors should be installed in the payload, such as radar, sonar or video imaging.

2.5 DSTO's Avatar Research Program

As part of an ongoing research program by the Defence Science and Technology Organisation (DSTO), the Codarra Avatar is being provided with an autonomous operation capability. The goal is not to develop a mission-ready UAV but to provide DSTO scientists with a research test bed suitable for studying the impact of providing UAVs with autonomy. The research program driving the UAV development focuses on two distinct areas. The primary focus is aircraft platform management: air and flight worthiness; health monitoring; and cost of ownership. A secondary research relates to autonomous software development issues: including validation and verification; architectures for coordinated autonomous behaviour; and autonomous controllers inspired by models of human cognition.

3. FLIGHT TEST

DSTO's Air Vehicle Division (AVD) in collaboration with the Air Operations Division (AOD) and the Department of Information Systems, The University of Melbourne, successfully conducted a flight test of the Codarra Avatar UAV between 5 - 7 July, 2004.

The specific mission, kept as simple as possible for this first set of trials, is to navigate to a specified waypoint, whereupon the agent will make a decision to turn to an alternate waypoint. After one of the alternate waypoints is intercepted, the UAV will then navigate to a recovery point. The decision about the alternate waypoint is based upon the agents perception of the local weather at the time of making the decision. For this reason it was not known prior to the mission what course the agent would choose to fly. The full flight-path of the mission is described and illustrated in Figure 1. This mission demonstrates, albeit in a simple manner, the capacity to fly a pre-briefed course and the capacity to exhibit autonomous decision-making based on local environmental factors.

Before the actual flight tests, a standard radio-controlled aircraft, similar in size and flight dynamics to the Avatar,

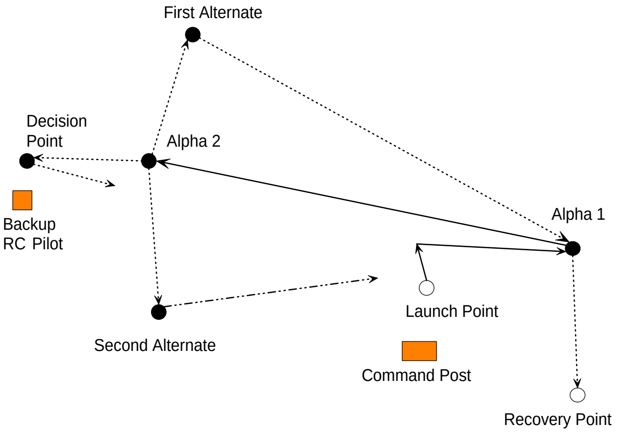

Figure 1: Avatar UAV Flight Path: At Alpha 2 the AUC (Autonomous UAV Controller) selects one of the alternate waypoints based on the local wind conditions. Failure of the AUC will result in the aircraft flying to the decision point and then returning home

was flown to test wind conditions and aircraft visibility. Then the team proceeded to setup the Avatar with waypoint data and conduct a 'walk around' test. This involved, essentially, switching on the FCS and JACK agent system, carrying the UAV around the flight course, traversing the waypoints the UAV is expected to intercept during its flight, and monitoring the FCS and JACK agent behaviour from the GCS. This testing method can be likened to hardware-in-the-loop testing, and might also be considered as a very low speed, low altitude flight test.

After testing was completed and minor configuration issues rectified, the flight test program was conducted. The same test was conducted two times: firstly with the iPAQ (housing the agent) *on-board* the plane (ie. directly connected to the FCS), and secondly with the iPAQ *off-board* and connected to the GCS (ie. communicating with the FCS over the GCS wireless link). The purpose behind the off-board test was simply to see if the agent platform can, in fact, operate off-board, thus allowing possibilities for more powerful processor platforms not restricted by size or power supply. Although the vision for the project is for the agent and FCS sub-systems to be both on-board the Avatar and have no dependance on radio communications to operate, investigating the off-board configuration was still useful.

The UAV was hand launched and manually piloted to a stable altitude and attitude by an expert radio-control (RC) pilot. It was then switched over to 'UAV mode', and proceeded to autonomously intercept the waypoints. All flight trials were safely and successfully conducted and an examination of the flight logs indicated that the agent had made the correct decisions at the correct times based on the data it received from the FCS.

Figure 2 shows some pictures taken during the test flight.

4. AGENT DESIGNS

4.1 Overview

The ultimate aim of our agent-based system is to achieve robustness and efficiency, but also for the design to be intuitive for designers and domain experts.

Figure 2: The Codarra Avatar during its test flight in Greytown, Melbourne, Australia, 5 - 7 July, 2004.

Two designs addressing the same mission parameters were concurrently developed. The "control systems approach" (§4.3) was developed to exhibit a simple, *bottom-up*, purpose built design, and as a means to compare to the "cognitive approach" (§4.4), which is a more sophisticated, *top-down* design, and was the system eventually used for the flight trials. In both designs, the agent-based mission management system sits on top of the control hierarchy, as will be described in the next section.

4.2 The Agent-FCS System Architecture

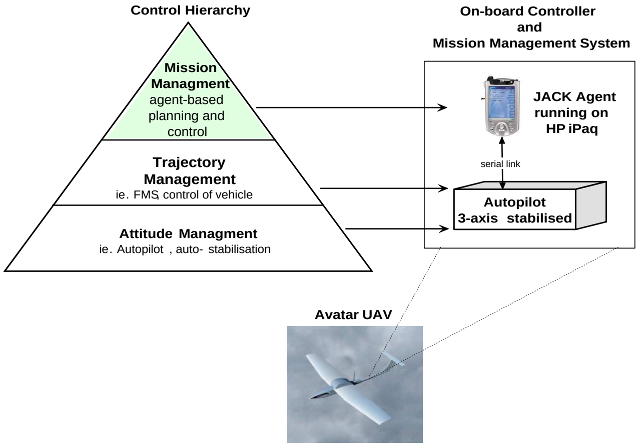
Control Hierarchy
On-board Controller and Mission Management System
Mission Managment
agent-based planning and control
Trajectory Management
ie. FMS control of vehicle
Attitude Managment
ie. Autopilot , auto- stabilisation
JACK Agent running on HP iPaq
serial link
Autopilot 3-axis stabilised
Avatar UAV

Figure 3: Agent - FCS Architectural design

Figure 3 [17] illustrates the system, and the respective hierarchical positions of the agent and FCS sub-systems within it. The FCS provides data about the state of the environment and the platform at 1 Hz. This data set includes GPS positioning data, control deflections, and aircraft state parameters. The FCS accepts many commands that provide the capacity to control the aircraft through waypoint setting or by interacting directly with the control surfaces. For the first trial only the waypoint commands were used. An interface was provided for the JACK agent to access data from the FCS. The agent was written in JACK, and hence operates inside a JACK Kernel that runs asynchronously with respect to the interface. This presented many synchronisation and integration issues.

4.3 Design 1: A Control Systems Approach

The first design that was implemented and tested in simulation was a simple, straight control-path design. The design's main premise is simplicity and minimalism, and follows a traditional feed-back control loop structure.

A brief description of the main design elements follows. A diagram of this design (consisting of JACK constructs) is shown in Figure 4.

Data triggered control mechanism As soon as data is presented to the agent in the form of packets, a linear control sequence is triggered. This sequence consists of, firstly, acquiring the data and putting it into the relevant JACK belief sets. Then, depending on the current situation (eg. intercepted a waypoint, reached a goal waypoint, etc.), actions are taken, which consist of either setting a new waypoint based on stored data, or performing calculations which will be later used to set waypoints.

Synchronous processing As there is a single, linear thread of control, complications and problems associated with asynchronous control are eliminated. However, for more complicated missions, asynchronous designs may be required from an implementation perspective, and desirable from a design perspective.

No unnecessary constructs As mentioned, the objective of this particular design is simplicity, and as a result many unnecessary modules were removed. Superfluous elements in a software system, in particular a real-time control system, are generally undesirable and should be avoided to ensure robustness and efficiency.

Immediate response Due to the simplicity of the design, which posts the minimal number of events and employs the minimal number of plans, cycle time is very short and hence results in a more reactive system.

This design is very efficient and robust for the purpose it was intended for. However, it is not flexible enough to handle different tasks outside of singular-decision type tasks (ie. tasks which only require one decision or action from the agent). This is obviously not ideal if a flexible, multi-purpose system is desired. However, according to the minimalist philosophy of robotics championed by researchers such as Brooks [5, 6], simple and purpose-built designs can sometimes be more effective, robust and consequently more powerful.

4.4 Design 2: A Cognitive Modelling Approach

The BDI language, JACK, already provides programming level constructs (belief, plan, event, capability) that map intuitively, if not entirely reliably, to everyday notions those constructs that are used by pilots to describe their mission control decision-making. A further architectural design layer was added that partitions the autonomous mission control software into four primary modules that map to the four elements of Boyd's OODA loop model [2], a model widely used in the military and is being adopted by businesses to model competitive activity. Each module reflects a human-oriented process. The model consists of four modules: Observe, Orient, Decide and Act, which are iteratively executed to perform the desired control.

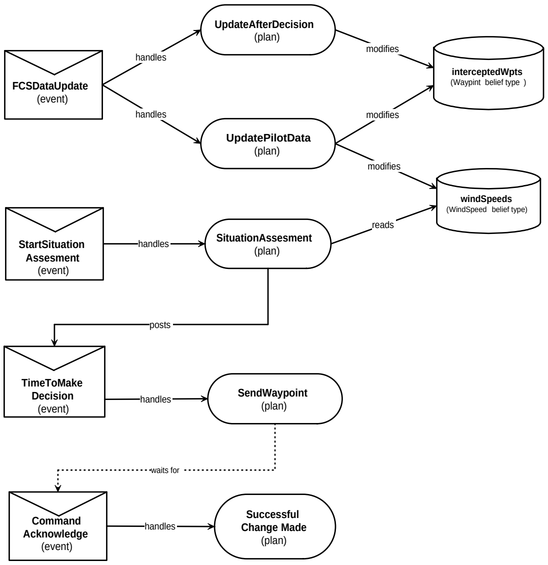

Figure 4: A Control System Design approach for the JACK AUC agent system. The main control loop triggers the `FCSDataUpdate` event, which starts the simple, linear control process of acquiring data and performing appropriate actions depending on the UAV's situation.

A detailed explanation of the functioning of these modules is beyond the scope of this paper but briefly the four primary modules are:

Observe Process the data arriving from the flight control system. Perform axes and units transformations, filter data searching for important events and relationships, trigger events, assert beliefs. The result of observation is an updated set of beliefs about the world and events to respond to.

Orient Process beliefs and observation events in the context of current intentions, mission goals, and beliefs. Construct *assessments*—reasoned context sensitive beliefs about the state of the world in the context of the current mission.

Decide From the assessments (and if necessary the observations) decide upon a course of action. The aim is to reason about appropriate actions from the more abstract view of the world provided by the assessments but the option to access the observations remains. A decision about an appropriate course of action might be considered to be the mission-level characterisation of the agents intent. An intent which is accorded detail in the following step.

[2] Sometimes the software modules are described synonymously as situation awareness, situation assessment, tactical selection, and standard operating procedures

Act Invoke the tactics that fly the aircraft. These plans provide the detailed pre-scripted recipes for achieving the desired mission.

Figure 5 graphically represents the architecture as it was implemented for the first round of flight-trials. The choice of architecture has the following properties:

Human Cognitive Model Based Architecture An architecture based on a human cognitive model allows the developer to leverage the folk-psychological familiarity and intuitiveness. It seems *obvious* where future functionality will reside and how the system integration will be performed. The design is simplified, although possibly at the expense of run-time complexity.

Asynchronous processing The software provides asynchronous processing between the modules in the OODA loop. The decision was taken because: of the modular nature of the OODA components; the *data-driven* nature of the design; and the potentially different temporal scales involved in the four OODA components.

Generation of abstract world state The design features an *orient* or *situation awareness* phase that generates a more abstract representation of the world state from the sensory data. This abstract world state simplifies decisions about tactics but requires that care is taken with the design.

Belief Triggered The design is largely data driven. In JACK terms it would be more accurate to describe the design as belief triggered. That is the belief data base causes the execution of most of the agent processing.

4.5 Discussion

It is noteworthy that the concepts and ideas for the control of UAVs presented in this paper are equally applicable to other unmanned vehicles, such as unmanned ground vehicles (UGVs) (eg. [3]) and autonomous underwater vehicles (AUVs) (eg. [7]).

The first design, inspired by a minimalist control systems approach, illustrates there may exist a simple and straightforward solution for many problems. The use of complex representations can, in many cases, be avoided and the system can consist of reactive, purpose-built elements. For the first design, the system *only* performs the task of navigating to a specific location and deciding on which direction to travel to. However, even small changes to the system can require a significant overhaul of the design and implementation. For example, if the UAV was required to navigate a shortest path (ie. path planning), then different modules for waypoint generation and command assignment, for example, would have to be developed. In such a case, the purpose-built nature of the original design may not be able to accommodate the new features/extensions and a completely new design may have to be developed.

The second design was inspired by *human surrogacy*, which is a notion that proposes that vehicle control systems can act, sense, behave and decide the way humans nominally do. The advantage of this design, based on the OODA loop, is that subtasks (ie. Observe, Orient, Decide, Act) are delineated by modules, and conform to good software engineering principles of strong cohesion, encapsulation, and low coupling. Moreover, the framework allows for extensibility and scalability. Depending on different tasks or missions, the modules can be modified to suit. Using the path planning example mentioned earlier, the OODA loop would be modified in the Decide and Act modules by adding the capability of generating and issuing waypoints to the FCS, rather than deciding upon predetermined waypoints as was the requirement in this first test flight.

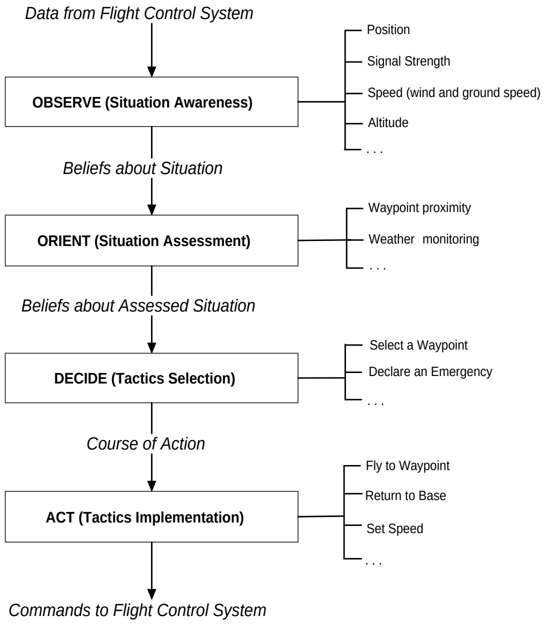
Data from Flight Control System
OBSERVE (Situation Awareness)
Position
Signal Strength
Speed (wind and ground speed)
Altitude
. . .
Beliefs about Situation
ORIENT (Situation Assessment)
Waypoint proximity
Weather monitoring
. . .
Beliefs about Assessed Situation
DECIDE (Tactics Selection)
Select a Waypoint
Declare an Emergency
. . .
Course of Action
ACT (Tactics Implementation)
Fly to Waypoint
Return to Base
Set Speed
. . .
Commands to Flight Control System

Figure 5: Cognitive Modelling (OODA) design approach for the autonomous UAV controller. The four primary modules implement a data-driven computational implementation of Boyd's OODA loop model of command decision making superimposed on the BDI language constructs of JACK.

In addition to flexibility, the OODA loop approach allows for effective *team cooperation*. Multiple UAVs forming a collective, or team, can all individually run OODA loops that interact via module interfaces. For instance, the *orient* module of an agent can be *observed* by one or more other agents, and the *action* of other agents can in turn influence the *decision* of the agent.

Further details of the lessons learned from the software design exercise are available in a forthcoming report [13].

5. RELATED WORK

This work borrows heavily from the experiences of developing agents for simulation described in §2.2, and from the general state of UAV controller design particularly those with a cognitive component, some of which are mentioned in §2.1. The *DyKnow* framework by Doherty and Heintz [10], for example, is essentially a signal-to-symbol transformer that, amongst other features, continually monitors perceived signals and creates higher-level *cognitive objects* that are relationally linked to each other. These abstract objects are continually updated in real-time by maintenance of predefined hypotheses that are defined by a domain expert at design time. The ideas were tested and illustrated using helicopter UAVs that were given the task of trying to interpret images taken from an on-board video camera, and correlate them with other sensor data such as GPS, altitude and velocity. Temporal information is also captured by creating chronologies of the cognitive objects. Specifically, the system was given the task of detecting cars that are either moving or stationary. The challenge addressed by the system was to effectively and efficiently *fuse* data from multiple sources that are distributed physically, semantically and temporally within the system execution cycle.

Wallis *et al.* [25] propose that BDI systems, and specifically UAV mission management systems, can be designed and developed without the need for 'direct' programming in agent languages such as JACK. Rather, a higher-level, visual development environment can be used that consists of a GUI with constructs familiar to the non-programmer (domain experts) designing the system. The visual development system maps to the underlying BDI implementation by precompiling the high level visual code. The system constructs consist of: goals, plans, world events and maintenance conditions; all of which are simplifications and generalisations of common JACK constructs.

An excellent paper by Clough [8] utilises the OODA loop and a human-focussed view of autonomy as the basis for a set of metrics that indicate the degree of autonomy of a UAV. Though as yet unattempted, it is likely that Clough's metrics could provide a suitable mapping of mission requirements (or level of autonomy) to the agent behaviours providing a starting point for agent development that meshes with the cognitively inspired agent designs described earlier.

6. CONCLUSIONS AND FUTURE WORK

This paper puts forward a *cognitively plausible* modelling approach to the control of unmanned vehicles. The presented idea moves away from traditional cognitive architectures based on symbolic representations (eg. Soar [18]), and elects to employ a simplified, cognitively plausible, process-oriented *OODA loop*, which is a model of cognition commonly used in military simulations.

Two distinct designs were implemented that have been developed and implemented in response to the same set of autonomous UAV control requirements. The designs were implemented in JACK, an agent-oriented language that supports BDI concepts. The first implementation and set of flight trials were kept deliberately simple, both to assist in making a more objective analysis on the ideas being studied, and to expedite the software development and flight testing activities, which were substantial for this first trial. The specific task or problem was for the UAV to make a single decision to intercept alternate waypoints given weather conditions and mission parameters.

Two designs were developed and compared. The OODA-based system (design two) is the focus of this paper, and was implemented for the first trials. We believe that using a cognitively plausible model for autonomous vehicle control, such as OODA, yields advantages in design intuitiveness, extendibility and scalability in autonomous vehicle controller design. The work presented here represents a starting point for further development. The first design, based on a minimalist feed-back control systems approach, illustrates that such designs can achieve comparable results, and reinforces the Brooksian philosophy of robotics. However, such an approach may lack the design intuitiveness, scalability and maintainability of more sophisticated systems, such as the OODA-based system.

Beyond the fact that an autonomous UAV controller based on a cognitive model was developed, there are lessons to be learned from the caveats and findings from the two approaches adopted. Future trials will examine the software required for UAVs to coordinate their activity in teams thus moving the research from single agent to multiagent systems. *Decentralised coordination*, and specifically designs built for decentralised coordination that are based on cognitive models, are areas for future consideration.

From a software design perspective some valuable lessons are already emerging. In our experiences from these experiments, developing UAV systems in terms of a cognitively plausible model assisted in the design and conceptualisation activities, and made the design more *intuitive*. In particular, we believe that designing *autonomous vehicles* that are "traditionally" controlled by human operators (such as unmanned helicopters, cars, submarines, etc.) in this way offers a level of *intuitiveness* that is a degree higher than traditional controller design. Designing extensions to the software also became intuitive and explanations of the design to non-technical stakeholders were facilitated. Though simpler to design and to conceptualise, the cognitive modelling version proved more difficult to debug and had more lines of code. Although the cognitive model has intuitiveness for some, engineers with experience in industrial development of controllers, autopilots, and autonomous systems may regard the use, in these applications, of a cognitive model as an unnecessary, even counter-productive burden. On the other hand, integrating domain expert knowledge (ie. pilot expertise, in the case of UAV controller design) into a system may also be beneficial, and developing such systems in terms of cognitive models may facilitate this objective. Considering these competing goals, the provision of autonomy for UAVs may ideally be seen as being on a spectrum between cognitive science and engineering with most successfully applied solutions likely occupying a middle ground.

It is as yet unknown whether the design expressiveness of a high-level agent language like JACK combined with an intuitive cognitive architecture like OODA will result in software that is easier to validate and certify. Expectations are that this might be the case and future work will explore this issue. Making use of more of the high-level features of JACK and layering a cognitive framework on top of the BDI architecture introduced an inevitable performance penalty. This penalty was tolerable on this trial (even at its most computationally intensive stages there was plenty of headroom on the iPAQ). Future flights will require greater computation but plans to move to dedicated single-board computers will increase the available resources. In resource bounded environments the computational overhead of a cognitive architecture might be prohibitive. Ongoing research will include considerations of payoff from trading design simplicity for run-time simplicity and for mechanisms for optimising the cognitive model.

An important consideration in the exercise of command and control is the transmission of *commander's intent*: a statement about the purpose for the activity and the expected end-state. It is often difficult to ensure that commander's intent remains widely promulgated between humans (for example in battle or in a business environment) and the problem is exacerbated by the introduction of autonomous entities. It is hypothesised that an agent langauge and software architecture that provides the possibility for explicitly representing statements of intent will offer command and control advantages. Future work will consider the implications for the explicit modelling of intent as part of the design of autonomous team based systems.

7. ACKNOWLEDGEMENTS

The first author is partially supported by a grant from the Australian Research Council (LP0211686) and Agent Oriented Software P/L (AOS). For assistance with system implementation, integration and flight trials, thanks to colleagues at DSTO - Shane Dunn, Steve van der Velden, Ian Powlesland; and from AOS - Richard Jones and David Shepherdson. From Melbourne University, thanks to Prof. Liz Sonenberg for her valuable input throughout the project.

8. REFERENCES

[1] Codarra Avatar fact page. Web: http://www.codarra.com.au/lowlpgs/1product.html, July 2004.

[2] Agent-Oriented Software Pty. Ltd. (AOS), P.O. Box 639, Carlton South, Victoria, 3053. *JACK Intelligent Agents^{TM}: JACK Teams Manual*, 4.1 edition, April 2004.

[3] R. C. Arkin and T. Balch. *Artificial Intelligence and Mobile Robotics*, chapter 11: Cooperative Multiagent Robotic Systems, pages 277–296. AAAI Press/MIT Press, 1997.

[4] C. J. Boyd. Observe-orient-decide-act (OODA) loop. An address given to 2025 participants, Air University, Maxwell AFB, October 1995.

[5] R. A. Brooks. A robust layered control system for a mobile robot. Technical report, Dept. of Computer Science, Massachusetts Institute of Technology, 1985.

[6] R. A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6(1&2):3–15, June 1990.

[7] D. Brutzmann, T. Healey, D. Marco, and B. McGhee. *Artificial Intelligence and Mobile Robotics*, chapter 13: The Phoenix Autonomous Underwater Vehicle, pages 323–360. AAAI Press/MIT Press, 1997.

[8] B. Clough. Metrics, schmetrics! how do you track a UAV's autonomy? In *Proceedings of the 1st AIAA Conference on Unmanned Air Systems*, Portsmouth, VA., USA, May 2002.

[9] B. Clough. Emergent behavior (swarming): Tool kit for building UAV autonomy. In *Proceedings of Swarming: Network Enabled C4ISR*, 2003.

[10] F. Heintz and P. Doherty. Dyknow: A framework for processing dynamic knowledge and object structures in autonomous systems. In *International Workshop on Monitoring, Security and Rescue Techniques in Multi-Agent Systems2004*, 2004.

[11] C. Heinze, M. Cross, S. Goss, T. Josefsson, I. Lloyd, G. Murray, M. Papasimeon, and M. Turner. Agents of change: The impact of intelligent agent technology on the analysis of air operations. In L. Jain, N. Ichalkaranje, and G. Tonfoni, editors, *Advances in Intelligent Systems for Defence*, volume 2 of *Series on Innovative Intelligence*, chapter 6, pages 229—264. World Scientific, River Edge, New Jersey, USA, 1 edition, December 2002.

[12] C. Heinze, S. Goss, T. Josefsson, K. Bennett, S. Waugh, I. Lloyd, G. Murray, and J. Oldfield. Interchanging agents and humans in military simulation. *AI Magazine*, 23(2):37–47, Summer 2002. An earlier version of this paper appeared in the Innovative Applications of AI conference, Seattle, 2001.

[13] C. Heinze, G. Murray, S. V. der Velden, I. Powlesland, and S. Karim. Lessons learned during the development of an autonomous mission controller for a small tactical UAV. Technical report, Air Operations Division, DSTO, 2005. in submission.

[14] C. Heinze, B. Smith, and M. Cross. Thinking quickly: Agents for modeling air warfare. In *Proceedings of the 9th Australian Joint Conference on Artificial Intelligence (AI'98)*, Brisbane, Australia, 1998.

[15] S. Karim, C. Heinze, and S. Dunn. Agent-based mission management for a UAV. In *Proceedings of the International Conference on Intelligent Sensors, Sensor Networks & Information Processing*, Melbourne, Australia, December 2004.

[16] M. Kovacina, D. Palmer, G. Yang, and R. Vaidyanathan. Multi-agent control algorithms for chemical cloud detection and mapping using unmanned air vehicles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.

[17] A. Lucas. First Flight - True UAV Autonomy At Last. *Agent-Oriented Software Press Release*, July 2004.

[18] A. Newell. *Unified Theories of Cognition.* Harvard University Press, Cambridge, Massachusetts, 1990.

[19] H. Parunak, S. Breukner, and J. Odell. Swarming coordination of multiple UAV's for collaborative sensing. In *Proceedings of Second AIAA "Unmanned Unlimited" Systems Technologies and Operations Aerospace Land and Sea Conference, Workshop and Exhibition*, San Diego, CA, USA, Sept 2003.

[20] M. Plehn. Control warfare: Inside the OODA loop. Masters thesis, Maxwell Airforce Base School of Advanced Airpower Studies, June 2000.

[21] A. S. Rao and M. P. Georgeff. BDI-agents: from theory to practice. In *Proceedings of the First Intl. Conference on Multiagent Systems*, San Francisco, 1995.

[22] N. Schurr, S. Okamoto, R. T. Maheswaran, P. Scerri, and M. Tambe. *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation.* Cambridge University Press, 2004.

[23] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

[24] M. Tambe, W.-M. Shen, M. Mataric, D. V. Pynadath, D. Goldberg, P. J. Modi, Z. Qiu, and B. Salemi. Teamwork in cyberspace: Using TEAMCORE to make agents team-ready. In *Proceedings of the AAAI Symposium on Intelligent Agents in Cyberspace*, pages 136–141, 1999.

[25] P. Wallis, R. Ronnquist, D. Jarvis, and A. Lucas. The automated wingman - using JACK intelligent agents for unmanned autonomous vehicles. In *Proceeding of Aerospace Conference, IEEE*, volume 5, pages 2615–2622, March 2002.