The OASIS Air Traffic Management System

August, 1992

Technical Note 28

By:

Magnus Ljungberg

Andrew Lucas

This paper will appear in the *Proceedings of the Second Pacific Rim International Conference on Artificial Intelligence, PRICAI '92*, Seoul, Korea.
This work is supported in part by the Civil Aviation Authority, Australia, the National Procurement Development Program of the the Department of Industry, Technology and Commerce, Australia, SUN Microsystems and by the Cooperative Research Centre for Intelligent Decision Systems.

Abstract

This paper describes research into automated support for air traffic controllers performing tactical air traffic management. It describes the OASIS system developed to help alleviate air traffic congestion. The system achieves this by maximizing runway utilization, achieved through arranging landing aircraft into an optimal order, assigning them a landing time, and then monitoring the progress of each individual aircraft in real time. OASIS is agent-oriented: its major components are independent agents, each solving a part of the overall problem. The system's flexible behavior results in part from this co-operative problem solving approach, and in part from the multiple levels of feedback employed between agents in the system and between the system and its environment. The highly dynamic nature of the air traffic control environment places very stringent real-time constraints on the system's reasoning processes, and requires the system to exhibit both goal-directed and reactive behavior. OASIS computes the landing sequence using an any-time algorithm. OASIS is currently implemented using the Procedural Reasoning System (PRS), a real-time reasoning system capable of reasoning about and performing complex tasks in a robust and flexible manner.

1 Introduction

Air traffic congestion is a major problem worldwide. It has been forecast that by the year 2000, the problem will cost almost US$10 billion annually in national losses due to constrained growth in Europe alone [Anon, 1990]. There are two principal approaches to alleviating this congestion: *demand management* and *capacity enhancement*. Demand management measures include slot allocation and landing fee surcharges during peak hours, and capacity enhancement can be achieved by measures such as constructing new runways. The high cost of capacity enhancement measures has caused increased interest in *air traffic management* for alleviating congestion and its associated delays.

Air traffic management modifies demand by controlling departures so that system capacity is not exceeded. Capacity is increased by providing efficient sequences of arriving and departing aircraft to minimize their inter-arrival time and thus eliminate idle time for the runway. In Australia the latter, known as tactical air traffic management, is performed by designated air traffic controllers called Flow Directors. The Flow Director regulates the air traffic flow by speeding up or slowing down aircraft due to land at an airport. The workload of the Flow Director is extremely high, and the job requires a very high level of skill and experience in all aspects of air traffic control.

OASIS (Optimal Aircraft Sequencing using Intelligent Scheduling) is a real-time artificial intelligence system developed to support the Flow Director. OASIS is developed as an application using the Procedural Reasoning System (PRS) [Ingrand *et al.*, 1992].

The design of OASIS is agent-oriented: the major components of the system are independent agents, each solving a part of the overall problem. The system's flexibility results from this co-operative problem solving approach.

The paper is organized as follows. The next section describes the tasks of the Flow Director in more detail. Section 3 introduces the requirements of an air traffic management system. Section 4 contains an overview of the system architecture. OASIS is being built using PRS, briefly described in Section 5. Section 6 contains an extended example illustrating the functions of OASIS, followed in Section 7 by a comparison of this system to others in the air traffic management domain. We conclude with a discussion of benefits of the OASIS design approach and by a description of the current status of the project.

2 Air Traffic Management

At major airports in Australia, a Flow Director sets up the landing sequence through the application of speed control and holding procedures, regulating the number of aircraft in the system when adverse weather conditions reduce capacity.

In essence, the Flow Director goes through the following process manually. On initial radar detection, the Flow Director estimates a possible landing time for the aircraft. The times are used to sequence the incoming aircraft into a landing order that avoids conflict and assigns them required landing times. Incoming aircraft must be sequenced so that the resulting rate of landings is not more than the maximum acceptance rate of the airport at that time. Usually the Flow Director establishes the landing time of an aircraft by the time it is within 120 nautical miles of the airport, or approximately 20 minutes from touch-down. The Flow Director determines these by choosing a strategy and asking each pilot to do one of the following things: to alter the speed of the aircraft; to modify the aircraft's path to the runway; or to perform airborne holding.

Once the landing sequence has been established and the pilots have been instructed to achieve their assigned landing times, the Flow Director then monitors progress. If an aircraft

appears likely to miss its landing time the Flow Director must take corrective action. This can be done by issuing new instructions to attempt to meet the original landing time, or by assigning the aircraft a new landing time, and possibly changing the landing order.

The Flow Director is *not* responsible for keeping aircraft safely separated from each other. This is a task that other air traffic controllers perform. However, the sequence, assigned landing times, and instructions issued by the Flow Director all have a direct bearing on the separation of the aircraft in the sequence.

At Sydney airport there can be as many as 60 aircraft landing per hour under favorable weather conditions. This rate is achieved by landing aircraft on two intersecting runways, and can be sustained for several hours. Other runway combinations result in an acceptance rate as low as 12 landings per hour.

3 Requirements of an Air Traffic Management System

There are a number of requirements for an air traffic management system.

The system must predict when an aircraft can be at the runway, considering past performance of that aircraft, its expected future performance, and the winds aloft.

The system must sequence each incoming aircraft into a landing order that maximizes the utilization of the runway resource. The sequence must be produced in a timely manner, and must reflect the current, changing, situation.

Once the system has assigned a landing time to the aircraft, it must generate a set of appropriate methods for achieving that time. The appropriateness and effectiveness of instructions depend on the aircraft's position. For example, airborne holding is only appropriate before the aircraft has passed the last holding point, after which only speed instructions and path shortening or stretching can be used.

The system should be capable of offering multiple alternative recommendations. In addition, Flow Directors must be able to override the system and issue their own instructions.

When an aircraft has been instructed to reduce speed or speed up to achieve a particular landing time, the system must monitor progress towards achievement of the intended landing time.

The system must react to changes in the environment, e.g. changes in wind direction, and also carry out prolonged chains of goal-directed reasoning, e.g. generating a set of instructions for achieving a landing time. It needs to balance the need for reactivity with this goal-directed behavior.

Next we describe the design of an agent-oriented architecture for air traffic management that fulfils these requirements. The design is based on the use of computational agents, each of which has its own beliefs, goals and commitments to plans, or intentions. That system, the Procedural Reasoning System (PRS), is described in more detail in Section 5.

4 Conceptual Design of OASIS

We have designed OASIS by sub-dividing the air traffic management task into its major parts and designing separate agents to solve each of those sub-problems. Each agent solves its part of the task independently, and co-operate with the others to produce the overall system behavior.

Agents communicate with each other and with the environment using messages. Messages are sent and received asynchronously, and are assumed to have assured delivery. However, no guarantees are given or assumed about the processing of the message once it has reached its recipient.

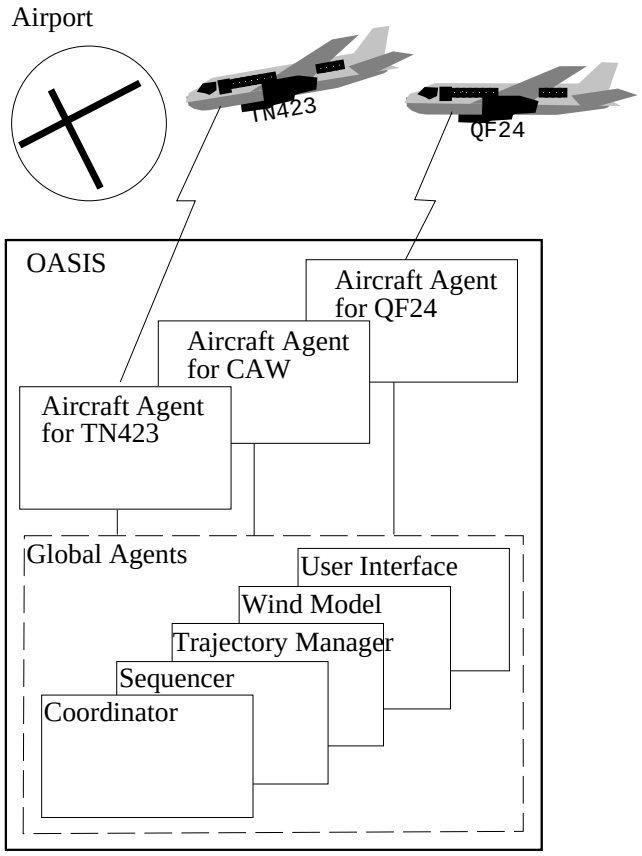

Figure 1: The structure of the OASIS system

Agents have integrity. Facts, goals, and intentions that are part of the internal state of the agent cannot be manipulated from the outside. For example, if an agent needs the co-operation of another agent, it must request that co-operation using messages. Moreover, the internal state of an agent is private. The only way for an agent to find out the belief, goal or intention of another agent is to send a message to that agent asking for the information.

As the environment changes, agents must decide how to act. If that deliberation continues for too long, the agent may find that the facts, goals, and state of the world on which the deliberation is based may no longer reflect the current situation. Hence, each agent must be able to reflect on the rate of change in the world and the tasks to be done, in order to effectively use its limited resources.

This design enables us to tailor each individual agent to the sub-problem it is solving. It allows for simplicity of design, high robustness and dynamically variable reactivity to external events. This *agent-oriented* design goes beyond the traditional subroutine concept or object-oriented design, as it depends crucially on each agent being an autonomous reasoner [Shoham, 1991].

OASIS is designed using two classes of agents. First, those that handle inter-aircraft co-ordination and reasoning, called *global agents*; and second, those that perform computation or reasoning relevant to each aircraft individually, called *aircraft agents*. Figure 1 shows how each aircraft has an agent associated with it, including those aircraft anticipated to arrive from flight plans and departure time messages, but not yet detected on radar.

There are five global agents. The COORDINATOR agent serves as the task manager, co-ordinating the activities of the other global and aircraft agents. The SEQUENCER agent

uses search techniques to arrange the aircraft in a least delay/cost sequence. The TRAJECTORY CHECKER agent verifies that instructions proposed by the system do not cause aircraft to violate statutory separation requirements. The WIND MODEL agent uses wind observations made by individual aircraft agents for predicting the wind field that aircraft are likely to encounter. Finally, the USER INTERFACE agent serves as the single point of communication with the Flow Director, managing all user interactions.

The system assigns an agent to each approaching aircraft intending to land at the airport. The AIRCRAFT agents contain all aircraft-specific data required by the system. The AIRCRAFT agent also assimilates position, speed, and altitude reports from real-time radar data. The tasks performed by an AIRCRAFT agent include estimates of when the aircraft will land, monitoring that the progress of the aircraft is as planned, and planning the trajectory of the aircraft.

In this paper we only give a detailed description of the SEQUENCER and AIRCRAFT agents.

4.1 The SEQUENCER Agent

The global SEQUENCER agent performs two essential tasks. First, it detects when congestion is likely. Second, it alleviates the congestion by assigning alternative landing times to the congested aircraft.

The detection of congestion is performed trivially by sorting the aircraft in landing time order and checking which aircraft are projected to land closer together than the minimum separation time.

Assigning new landing times to the aircraft is done by arranging the aircraft into a minimal delay sequence using an A^*-based search procedure [Nilsson, 1980]. There are a number of constraints on the problem, listed below.

The availability of runway resources. A runway may have different capacities at different times. For example, during instrument landing conditions the acceptance rate will be lower than during visual conditions.

The minimum inter-arrival time between aircraft. This depends on the types of the aircraft. For example, a light aircraft following a heavy aircraft requires greater separation between them than a heavy aircraft following a light aircraft. Heavy aircraft generate more wake turbulence than light aircraft, and light aircraft are more sensitive to wake turbulence than heavier aircraft.

Aircraft have limited endurance, and they cannot wait indefinitely to land as they carry limited amount of fuel. In cases when fuel is short, an aircraft will divert to an alternative airport.

Aircraft have limited performance envelopes. They have maximum and minimum speeds which often severely restrict the range of landing times achievable.

The objective is to minimize some delay-related cost function, for example, the total delay weighted by the priority of the aircraft.

The search space for the sequencing problem is fairly large, and the time constraints imposed by the environment are such that we have been forced to accept sub-optimal sequences under certain circumstances. We allow the optimization process to be interrupted when a solution is required, but before the algorithm has found an optimal one. When it is interrupted it returns the lowest-cost solution found so far. If there are aircraft not included in that sequence, they are added to the end of the sequence in a first-come first-served order. The SEQUENCER agent uses an *any-time* algorithm [Boddy and Dean, 1989], that is the cost of the sequence produced decreases monotonically to the cost for the optimal solution as a longer time is allowed for the sequencing process to run.

4.2 The AIRCRAFT agent

The database of the AIRCRAFT agent contains the flight plan-related information of the aircraft, a performance model for that aircraft, and its general state (such as whether the aircraft is climbing, descending, or holding in a holding pattern).

An AIRCRAFT agent has three principal components: the PREDICTOR, the MONITOR, and the PLANNER. The task of the PREDICTOR component is to estimate when an aircraft will be at different points along its trajectory and, in particular, at the destination airport. The MONITOR compares those predictions to the actual aircraft arrival times, and notifies other OASIS agents if there are substantial discrepancies. Finally, the PLANNER accepts a landing time assigned by the SEQUENCER, and plans how to achieve that landing time. It then offers an appropriate set of plans to the Flow Director and adopts the ones selected for each aircraft agent.

The PREDICTOR Component

The PREDICTOR component computes when the aircraft can reach an airport. The parameters to this prediction process include the nominal speed, the nominal altitude, and the wind velocity. We have developed a library of nominal aircraft performance profiles derived from published performance data for a range of aircraft types, this data describes the speeds and altitudes as functions of the distance traveled from the departure point and the remaining distance to the destination airport. Aircraft agents select performance characterizations depending primarily on aircraft type but we also make provisions for different airline operating procedures. The wind velocity information is requested from the WIND MODEL agent, using the flight plan to specify for which geographical area the information is needed. The SEQUENCER agent requires not only the possible landing time, but also a performance envelope within which the aircraft can be sequenced. That performance envelope is produced by the PREDICTOR computing the arrival time with a maximum speed profile and a minimum speed profile. The results of the PREDICTOR are used for three purposes: by the SEQUENCER agent as a basis for assigning a landing time; by the MONITOR component for future wind velocity estimates; and by the PLANNER component to plan the potential effects of future instructions to the aircraft.

The MONITOR Component

The MONITOR compares the predictions made by the PREDICTOR with the actual aircraft arrival times at different points along its trajectory. The MONITOR notifies the rest of the system if the actual performance falls sufficiently far outside the limits of what must be achieved to meet the assigned landing time.

The four major reasons for a discrepancy between the predicted time and actual time of arrival are that the aircraft may not have: followed the planned path; held the planned altitude; held the planned air speed; or the wind may have been different from the estimate used when computing the prediction.

The MONITOR detects the first two conditions directly from radar data. The latter two are much harder to detect. The system must compare the performance of several different aircraft before it can distinguish between aircraft not holding their expected air speeds and wind. The MONITOR therefore sends its observations to a global agent, the WIND MODEL agent, that is designed to perform this global analysis.

In summary, observations made by the MONITOR component are used for three purposes: to correct the behaviour of *this* aircraft; to improve future predictions of times for *this* aircraft; and to improve predictions of landing times for *subsequent* aircraft. These tasks

are done by the PLANNER and PREDICTOR components of the AIRCRAFT agent and by the WIND MODEL agent, respectively.

The PLANNER Component

The PLANNER component of the aircraft agent constructs plans that cause aircraft to land at their assigned landing time. The global SEQUENCER agent computes the landing times for all aircraft. The PLANNER is responsible for producing plans of *how* the desired landing time is met. A plan in this context is the future trajectory of the aircraft, the air speed profile and the altitude profile. The PLANNER changes these three in concert to cause the aircraft to land at its assigned landing time. The plan is eventually be conveyed to the pilot as changes to aircraft path, altitude, or air speed.

The PLANNER has to operate with the following constraints: (1) the plan must conform to guidelines regarding avoidance of impracticable instructions. For example, Flow Directors do not normally issue a delaying instruction after having asked an aircraft to speed up; (2) the plan must suggest speed instructions in increments of 10 knots; (3) the plan must be within the performance envelope of the aircraft; and (4) the plan must be compatible with airport procedures, for example, with respect to noise abatement procedures, curfews, etc.

The PLANNER has a library of various strategies for forming appropriate plans. Some strategies are based on strict algorithmic models, while others are heuristic. The PLANNER selects a suitable method according to the current situation.

When the PLANNER has generated a set of plans, it gives the Flow Director the choice of which plan to adopt. When a plan is adopted and the corresponding instructions have been issued to the pilot, the PLANNER enters a different mode of operation. It now attempts to land the aircraft as closely as possible to the assigned landing time. If the MONITOR component notices a large time discrepancy, the PLANNER first attempts to meet the assigned landing time by altering the originally adopted plan. If the discrepancy is too large, that is there is no possible plan for achieving the assigned landing time, the PLANNER ask the SEQUENCER for a new landing time.

For example, an aircraft, `A`, can land at 01:02 unless directed otherwise. However, it has been assigned a landing time of 01:08, a delay of 6 minutes. This delay can be achieved by two means, either speed reduction or holding. OASIS would offer the following choice to the Flow Director:

```
To land A at 01:08, a delay of 6 minutes:
(1) Decrease speed from 300 to 230 knots; or
(2) Hold at BIK for 6 minutes, until 00:57.
```

This design approach allows us to introduce multiple levels of nested feedback between agents. For example, the MONITOR component feeds back the observed performance to the PREDICTOR and PLANNER components, and to the WIND MODEL. It gives OASIS the following three properties: (1) Reactivity – a minor time discrepancy is corrected by additional instructions by the PLANNER component; (2) Good local prediction – if the discrepancy is due to wind or to the pilot operating the aircraft consistently outside its assumed performance profile, the PREDICTOR provides better future estimates of the landing time; and (3) Good global prediction – if a pattern of discrepancies can be observed for aircraft in the same geographical area, the global WIND MODEL is able to suggest better wind effect estimates for aircraft in the same area.

5 Procedural Reasoning

PRS is designed to be used as a situated real-time reasoning system. As shown in Figure 2, PRS has (1) a *database* containing current *beliefs* or facts about the world; (2) a set of current *goals* to be realized; (3) a set of *plans* describing how certain sequences of actions and tests may be performed to achieve given goals or to react to particular situations; and (4) an *intention structure* containing all plans that have been chosen for execution. An *interpreter* manipulates these components by selecting appropriate plans based on the system's beliefs and goals, placing those selected on the intention structure, and executing them. The system interacts with its environment through its database and through the basic actions that it performs when its intentions are carried out.

5.1 Goals and Beliefs

The beliefs of PRS provide information on the state of the environment. They are represented in a first-order logic. For example, the fact that the position of an aircraft called `CAW` is latitude $-33.945°$ and longitude $151.172°$ can be represented by the statement `(position CAW -33.945 151.172)`.

The goals of PRS are descriptions of desired tasks or behaviors. In the logic used by PRS, the goal to achieve a certain condition `C` is written `(! C)`; to test for the condition is `(? C)`; and to wait until the condition is true is `(^ C)`. For example, the goal to decrease the speed of an aircraft, `CAW`, to 150 knots could be expressed as `(! (reduced-speed CAW 150))`; to test for it could be expressed as `(? (reduced-speed CAW 150))`; and to wait for the condition to become true could be expressed as `(^ (reduced-speed CAW 150))`.

5.2 Plans

Knowledge about how to accomplish given goals or react to certain situations is represented by plans in PRS (see for example, Figure 3). Each plan has a *body*, which describes the steps of the procedure, and an *invocation condition*, which specifies in what situations the plan is useful and applicable [Georgeff and Lansky, 1986].

The body of a plan can be viewed as a plan schema. It is represented as a graph with one distinguished start node and (possibly multiple) end nodes. The arcs in the graph are labeled with the subgoals to be achieved. Successful execution of a plan occurs when all subgoals on a path from the start node to an end node are successfully achieved. This formalism provides a natural and familiar representation of plans involving the usual control constructs, including conditional selection, iteration, and recursion.

The invocation condition describes the *events* that must occur for the plan to be executed. Usually, these events consist of the *acquisition* of some new goals (in which case the plan is invoked in a goal-directed fashion) or some *change* in system beliefs (resulting in data-directed or reactive invocation) and may involve both.

The set of plans in a PRS application system not only embodies procedural knowledge about a specific domain, but also includes *metalevel* plans; that is, information about the manipulation of the beliefs, goals, and intentions of the system itself. For example, typical metalevel plans encode various methods for choosing among multiple applicable plans. They may also modify and manipulate intentions, computing the amount of reasoning that can be undertaken, so as to act within the real-time constraints of the problem domain.

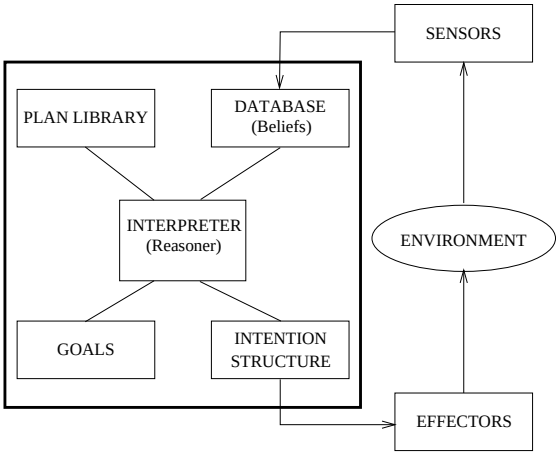

Figure 2: The structure of the Procedural Reasoning System

5.3 The Intention Structure

The intention structure contains all plans the system has chosen for execution, either immediately or at some later time. The adopted plans are called *intentions*. A single intention consist of a top-level plan together with all the sub-plans that are being used in attempting to carry out the plan.

At any given moment, the intention structure may contain several such intentions, some of which may be suspended or deferred, some of which may be waiting for conditions to hold, and some of which may be metalevel intentions to choose between various alternative plans.

5.4 Execution

Unless some new belief or goal activates a new plan, PRS tries to fulfill any intentions previously decided upon. This results in focused, goal-directed reasoning in which plans are expanded in a manner analogous to the execution of subroutines in procedural programming systems. However, if some important new fact or goal becomes known, PRS re-assesses its intentions. It may, for example, suspend or terminate its current task and choose to work on something else. Thus, not all options that PRS considers arise because of means-end reasoning. Changes in the environment may lead to changes in the system goals or beliefs. These changes may in turn result in the consideration of new plans that are not means to any already intended end. PRS can therefore change its focus completely and pursue new goals when the situation warrants it.

5.5 Multiple Systems

In this co-operative agent-oriented application, it is necessary to monitor and process many sources of information simultaneously. PRS was designed to allow several interpreters to run in parallel to achieve this behavior. Each interpreter has its own data base, goals, and plans, and operates asynchronously to other interpreters, communicating with them by sending messages.

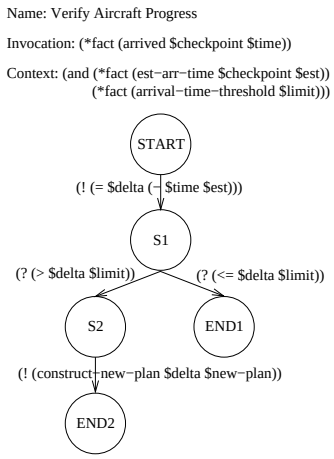

Figure 3: Plan for reacting to an aircraft arriving at a checkpoint

6 Sample Situation

To highlight some important features of OASIS, we examine an extended example. An airport operating with one runway and a minimum inter-arrival time of 3 minutes. There are three aircraft, aircraft `S1` and `S2` arriving from the south, and aircraft `N` arriving from the north. All three aircraft have the same level of priority, and are of the same type, e.g. Boeing 737-300. Aircraft `S1` and `S2` are following the same path: from a second airport ML, via a radio beacon named BIK, to the destination airport SY. Aircraft `N` departs a third airport BN, and travels to airport SY via the radio beacon BIK. The aircraft are detected on the radar when they are approximately 180 nautical miles from the airport, about 27 minutes from touchdown. Their predicted landing times are shown in Figure 4.

OASIS sequences aircraft `S1` to land first at 01:00, one minute ahead of its unrestricted landing time; aircraft `N` second at 01:03, one minute after its unrestricted landing time; and aircraft `S2` third at 01:06, three minutes after its unrestricted landing time. The total delay for this sequence is four minutes: aircraft `S1` is one minute ahead of its landing time, that is no delay, aircraft `N` is delayed by one minute, and aircraft `S2` is delayed by three minutes. An alternative sequence is aircraft `S1` first at 01:01, the same landing time as its unrestricted landing time; aircraft `N` second at 01:04, two minutes after its unrestricted landing time; and last aircraft `S2` at 01:07, four minutes after its unrestricted landing time. This sequence has a total delay of 6 minutes. OASIS chooses the sequence with the lowest total delay, that is the first sequence. The SEQUENCER agent then distributes the assigned landing times

Aircraft	ETA at BIK hh:mm	ELT for SY hh:mm	Earliest hh:mm	Latest hh:mm
`S1`	00:50	01:01	01:00	01:30
`N`	00:49	01:02	01:01	01:40
`S2`	00:52	01:03	01:02	01:40

Figure 4: The estimated time of arrival (ETA) at the radio beacon BIK, the estimated landing (ELT) time for the airport SY. In addition, the earliest and latest possible landing times allowed by the performance envelopes of the aircraft are also shown. These estimates are made when the aircraft are initially detected on radar, assuming unrestricted flight to the airport.

Aircraft	ALT hh:mm	Delay minutes	Speed knots	Path Change nautical miles
S1	01:00	−1	320	−
N	01:03	+1	280	+7.5
S2	01:06	+3	260	+23

Figure 5: The assigned landing time, ALT, for SY, the time to make up or lose, and the alternative flow control instructions for achieving this for each aircraft

to the individual aircraft agents: S1 at 01:00; N at 01:03; and S2 at 01:06. The PLANNER components of each aircraft agent receive these messages, and start planning how they can achieve their assigned landing times.

The PLANNER of aircraft agent S1 has to choose between either directing its aircraft along a more direct route or by increasing the air speed. To gain one minute by path control it would have to shorten the path by at least 7.5 nautical miles. As the track to the airport being followed by aircraft S1 is essentially radial, the path cannot be shortened sufficiently. The other alternative, speed increase, is also investigated. In order to gain one minute in the descent phase, the speed needs to be increased from the nominal speed of 300 knots to the maximum descent speed of 320 knots.

The PLANNER for aircraft agent N can choose between three alternatives for achieving its landing time: it can extend the path; it can decrease the descent speed; or it can hold the aircraft. However, holding an aircraft normally requires at least four minutes, so that option can be ruled out. The PLANNER eventually recommends either extending the path in cruise by 7.5 nautical miles, or decreasing the descent speed from the nominal 300 knots to 280 knots. The Flow Director is then presented with the choice of the two instructions. The aircraft agent accepts the instruction selected by the Flow Director, and changes either the path of the flight plan for aircraft N, or its expected future descent speed. The PLANNER of the third aircraft agent performs a similar reasoning and suggests either a increased path by 23 nautical miles, or a decreased descent speed of 260 knots. Figure 5 contains a summary of the suggested instructions.

Assuming that the Flow Director selects the speed control instruction for all three aircraft, the revised arrival times are shown in Figure 6.

As each aircraft passes the radio beacon BIK, the MONITOR in the aircraft agent compares the actual time of arrival with the predicted one. In this example, let us assume that aircraft N arrives at BIK at 00:51, one minute after the predicted time. The MONITOR notices the discrepancy and communicates it to the PLANNER component. The PLANNER chooses to adjust the speed instruction to increase the descent speed back to the nominal 300 knots, bringing aircraft N onto the runway at the initially assigned landing time, 01:03. Had the discrepancy been larger than that gained by cancelling the initial instruction, the PLANNER of the aircraft agent N would have had to ask the global SEQUENCER agent for

Aircraft	ETA at BIK hh:mm	ELT for SY hh:mm
S1	00:49	01:00
N	00:50	01:03
S2	00:55	01:06

Figure 6: The revised estimated time of arrival (ETA) at the radio beacon BIK, and the revised estimates of landing time for SY for each aircraft

a new, later landing time that could feasibly be achieved. This would have caused `S2` to be assigned a new landing time also, in order to maintain the minimum separation time.

7 Other Systems

There are three other developments worth considering. Two are currently in operation, the third is undergoing initial trials. The first, COMPAS, is a flow control and metering system developed for the German Civil Aviation Authority (BFS) by the German aerospace research centre DLR. Operational use of COMPAS commenced in Frankfurt Airport in 1989. COMPAS is a conventional computer program, written in FORTRAN and C. COMPAS only considers single runway sequences. It does not suggest detailed instructions to the Air Traffic Controller; and once the sequence has been set, progress of the aircraft involved in the sequence is not automatically monitored [Schubert, 1990].

The second, MAESTRO, was developed by CENA, the research organization of the French Civil Aviation Authority. MAESTRO is used operationally for the domestic airport at Orly, south of Paris [Garcia, 1992]. MAESTRO is similar to COMPAS in that it only considers one runway. However, it does monitor the progress of individual aircraft, and indicates to the user the progress made against the plan. MAESTRO has also been developed as a conventional real-time system, written in ADA.

The third, CTAS, is under development at NASA-AMES for the U.S. Federal Aviation Agency. It is far more complex than either COMPAS or MAESTRO. The core of CTAS is performance computation. CTAS is a conventional system, but distributes functionally between the different air traffic control positions in the radar control centers. CTAS has been installed at Denver for initial trials.

8 Conclusions

The central design principle of OASIS is the functional distribution of tasks into autonomous, co-operating agents. This design methodology has yielded major benefits. The design has deliberately treated the major components as black boxes. The aim is to not let any individual agent rely on the *accuracy* of any other agent in the system.

The design is inherently robust. If an individual aircraft agent fails, other aircraft agents continue to function. If any functional part of an aircraft agent fails, it does not cause failures in any other aircraft agents or global agents. Moreover, the accuracy of other agents is also be maintained at the same level as before the failure occurred, unless they need to interact with the failed agent.

The design has multiple levels of nested feedback loops between the aircraft agents and the environment. These methods of feedback provide OASIS with high reactivity and an ability to use wind estimates locally and globally for improving landing time predictions.

OASIS is implemented in a simulated air traffic environment on Unix workstations. The PRS real-time system is currently implemented in LISP, but will soon be replaced as the system kernel by a C++ based system (MARS). OASIS has been under development for 2.5 years. It currently handles realistic peak hour traffic samples from Sydney airport, involving 65 arriving aircraft over a time period of 3.5 hours.

Acknowledgments

The development of OASIS has been a team effort involving Noel Karppinen and Peter McLeod from the CAA, Philippe Repusseau, Andrew Hodgson and Paul Maisano from AAII. The authors would like to thank Michael Georgeff, David Kinny, Liz Sonnenberg and Anand Rao for reviewing drafts of this paper.

References

[Anon, 1990] Anon. A European planning strategy for air traffic to the year 2010, 1990. Prepared for: International Air Transport Association Geneva/Montreal.

[Boddy and Dean, 1989] M. Boddy and T. Dean. Solving time-dependent planning problems. In *Proceedings of AAAI-89*, Los Altos, CA, 1989. Morgan Kaufmann Publishers,.

[Brooks, 1985] R. A. Brooks. A robust layered control system for a mobile robot. Technical Report 864, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, 1985.

[Garcia, 1992] J-L Garcia. MAESTRO–a metering and spacing tool. Technical report, Centre d'Etudes de la Navigation Aérienne, Orly Sud, France, 1992.

[Georgeff and Lansky, 1986] M. P. Georgeff and A. L. Lansky. Procedural knowledge. In *Proceedings of the IEEE Special Issue on Knowledge Representation*, volume 74, pages 1383–1398, 1986.

[Ingrand *et al.*, 1992] F. F. Ingrand, M. P. Georgeff, and A. S. Rao. An architecture for real-time reasoning and system control. *IEEE Expert*, forthcoming, 1992.

[Nilsson, 1980] N. J. Nilsson. *Principles of Artificial Intelligence*. Tioga Publishing Company, Palo Alto, CA, 1980.

[Schubert, 1990] M. Schubert. COMPAS system concept. In *The COMPAS SYSTEM in the ATC Environment*, Braunschweig, Germany, 1990. DLR Institute for Flight Guidance.

[Shoham, 1991] Y. Shoham. Agent-oriented programming extended abstract. In *Proceedings of the IJCAI-91 Workshop on Theoretical and Practical Design of Rational Agents*, 1991.

[Steep *et al.*, 1988] A. Steep, S. Carmarata, F. A. Hayes-Roth, P. W. Thorndyke, and R.B. Wesson. Distributed intelligence for air fleet control. In H. A. Bond and L Gasser, editors, *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann Publishers, San Mateo, CA, 1988.