SIMULATION OF A CONFLICT MANAGEMENT SYSTEM FOR AIR TRAFFIC CONTROL

Baird Ndovie

DAKE Centre,
University of Keele, Keele,
Staffordshire, ST5 5BG, U.K.
e-mail: baird@cs.keele.ac.uk

ABSTRACT

The Air Traffic Control (ATC) domain is concerned with controlling and monitoring air traffic from a source airport to a destination airport. A number of cooperating agents distributed throughout the ATC environment ensure that a safe, orderly and expenditious movement of traffic in airspace is achieved.

Starting with a general discussion on the mechanics of ATC especially its operation, the paper develops an ATC model. The model addresses two issues:

1. *A model of an agent which will allow interagent cooperation.*
2. *A model of an operational environment which requires interagent cooperation.*

The paper terminates with a simulation of an ATC system, known as a Conflict Management System (CMS), to be implemented. CMS depicts interagent cooperation among air traffic controller (ATCO) agents, and between ATCO agents and aircraft agents during flight execution.

1 INTRODUCTION

Air Traffic Control (ATC) is applied to a variety of Civil and Military traffic in circumstances which range from the formalised progression of scheduled passenger services along airways, through the relatively unstructured movements of military sorties involving low level missions[6]. Wherever control is exercised, however, the common requirements are to give guidance to the destination, to maintain safety, and to manage airspace and runway resources. Therefore, the purpose of ATC is to ensure a *safe, orderly and expeditious* flow of air traffic from point of origin to destination. With all the sky to choose from, one would think it is easy in achieving these objectives. In practice,

there are a number of constraints which mean that expedition and orderliness frequently have to be sacrificed to the paramount requirement of safety. Some of the constraints are functions of the way that aircraft fly, how airlines operate them, the routes aircraft have to follow, the weather, etc. The general effect of these is to produce a demand for quite a large number of aircraft to fly along the same airways, at the same restricted range of height, at the same times of day and taking off from and bound for the same airports capable of handling them. The result is a situation analogous to a highway which ends in a bottleneck. This situation would be hazardous without control.

Individual air traffic controllers are normally organized into a system in which each controller has a zone of responsibility and a number of designated aircraft to control. Controllers thus operate as a serial system, the aircraft being passed from one controller to the next via a formalised handover mechanism. A controller receives input flow as aircraft to which he allocates airspace and passes output flow of aircraft to the controller responsible for the next stage of flight. Information on status of aircraft and resources is fed forward and backward between controllers to provide a basis for future planning. Sometimes changes in status initiate replanning.

A system, such as air traffic control, involves many technologically advanced components. It seems to be a good choice for a high degree of automation. However, the present level of automation in any ATC system in the world is minute; the human controller who is the integral part of the system, not only makes the decisions himself but also carries out a great deal of mental processing. Such a conservative approach is due to the high level of safety demanded of the system.

Completely computerised controlled system have yet to show themselves capable of providing such integrity, which has led to many researchers in DAI [1, 2, 13, 14, 9] and CKBS [3] to investigate the feasibility of fully automating the current air traffic control system. Computers in current ATC system simply manipulate data for presentation to air traffic controllers. We visualize, in a fully automanted ATC system, that a computer will be characterised by its ability to analyze data and determine what actions are required to achieve a desired system state such as maintaining a safe, orderly and expenditious flow of traffic. Air traffic controller will then be responsible for determining that the automated system is functioning properly, for handling exceptions and in case of system failure to coordinate backup plans.

Our objective in this research is to apply CKBS techniques and methods to ATC system design. We are especially concerned with modelling how controllers (refered to as *Agents*) detect and resolve conflicts at the time of flight execution [12].

In this paper, we discuss the design, simulation and implementation issues of an ATC system. Section 2 presents an overview of air traffic control by describing its mechanics. Section 3 presents our ATC model, describes the structure and types of agents in the system, contents of the agents together with data structures and operations suitable and meaningful to ATC environment. The model of the operational environment

is discussed in section 4, and section 5 describes briefly the part of ATC system we are attempting to simulate and implement. We end with a conclusion in section 6.

2 AIR TRAFFIC CONTROL OVERVIEW

Air traffic control(ATC) started due to two conditions which dominated scheduled airlines in the early years of aviation. Bad weather forced aircraft to fly without visual reference to the ground or other aircraft in the vicinity; and the growth of competitive aviation caused aircraft to use the same routes to the same destination at the same time. In the past, air traffic controllers used to stand beside the runways signalling landing and departing aircraft with flags. Later they moved to control towers with radios instead of flags. As air traffic increased, the job of a controller became a more specialized one [4].

Air traffic, nowadays, fly under the provision of Instrument Flight Rules (IFR). A system of navigational beacons was set up with routes, called airways, between these beacons. Aircraft wishing to fly in bad weather generally fly along these airways and are controlled by numerous air traffic controllers along the way. The best way to understand the ATC system is to follow a hypothetical flight from one loading bay to the next. An aircraft files its flightplan to the control tower for verification. When ready to taxi, it calls *ground* control, which is responsible for coordination of all aircraft on the airport's surface except that on the active runway. With ground control's guidance, the aircraft moves to the edge of the active runway and is handed off to *tower controller*.

A tower controller, who is responsible for visual control of all traffic within a radius of about five miles, clears an aircraft for take off. A tower controller hands off the aircraft to a *departure controller* who tracks aircraft within a radius of about twenty-five miles using radar. The aircraft's identification code, altitude, speed and other relevant information are shown on radar display. The departure controller hands off aircraft to the first series of *enroute controllers*. These controllers monitor aircraft in a more or less cruising flight level with radar, resolving minute by minute situations which might affect aircraft safety. Each controller is responsible for a portion of airspace, called a sector. Control of aircraft is passed from one sector to the next as the flight proceeds. As it nears its destination, it is handed to an *approach controller* whose function is same as of a departure controller, except that he is concerned with arrivals instead of departures. When visual contact has been established with airport, the tower controller and the ground controller follow the aircraft safely to the parking area. This entire sequence is shown in figure 1.

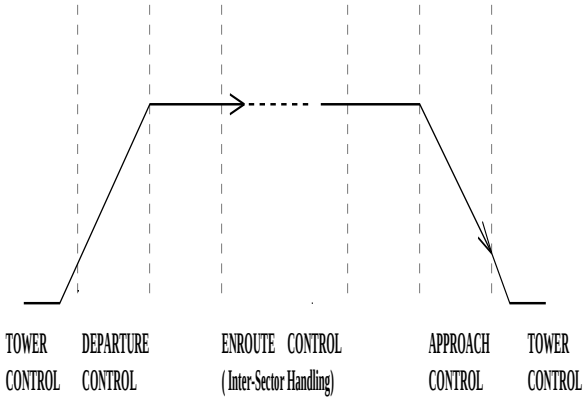

Figure 1: Motion of Aircraft

From take-off to touchdown, aircraft experience a variety of conflicts which can be categorised as *time*, *separation* and *handover* conflicts. In the ATC environment, the main resource to be shared among aircraft is airspace and conflicts usually occur when there is a high density of traffic in the airspace. In such a situation, two or more aircraft may possibly violate the statutory separation distances. The conflicts are resolved by performing the following actions on aircraft: change of route of aircraft, regulation of speed, change of direction, change in flight level and holding of aircraft in a circular state. A successful conflict resolution means that safety among aircraft is achieved and that constraints, such as getting to the desired destination with minimum fuel consumption, are satisfied.

2.1 Operations of air traffic control

Air traffic control is a command and control system which is fairly complex, and consists of a number of different components (e.g Controllers, Aircraft, Airports, etc). Traffic is controlled through a set of formal procedures [11].

As mentioned in the previous section, an ATC system is triggered when an aircraft requests approval of a flighplan from an air traffic controller who issues permission to fly the requested route. As aircraft travel along the approved flightpaths, there is independent surveillance provided through ground based radars and transponders on board aircraft. Surveillance includes aircraft identification and position. This information is relayed to an air traffic controller who monitors the progress of aircraft along the approved flightplans. A controller then generates commands that he transmits to aircraft to ensure conflict-free and expenditious flight along the route specified in the flightplan. The air traffic controller is obligated to accept and act on aircraft requests and to provide traffic and environmental advisories, weather, etc. When aircraft passes from one controller's jurisdiction to another (that is, from one sector to the next), there is a formal handover procedure.

The system has as its inputs information regarding aircraft's intended flightpath and progress along that path. This information is updated at scheduled intervals and printed on flight progress strips (discussed later). Knowing what an aircraft intends to do versus what it is doing allows a controller to issue effective instructions and maintain safety by looking ahead in time for potential conflicts. The other source of data is information about nationwide traffic level, airport capacities and weather. The system operates continously for a given aircraft until some completion criteria is specified such as an aircraft has left the controlled airspace by landing or by flying beyond the controlled airspace.

3 MODEL OF AIR TRAFFIC CONTROL

The analysis above provides the basis for our model of air traffic control in which air traffic controllers (which are agents) and aircraft (another agent) cooperate among each other to achieve their goal (ie, safety in airspace). Agents in our model perform tasks. Each task is performed by making use of other agents in the system. An agent negotiates with other agents and proposes a solution to any problem presented to it. In an ATC model, agents detect and resolve conflicts which occur in airspace. A controller agent detects conflicts by evaluating flightplan or flight progress information received from other controller agents. An agent resolves a conflict by negotiating with other agents and propagates a resolved conflict to all appropriate agents. To do that, agents receive messages from other agents, provide and transmit responses to other agents. Hence, in our opinion, our model needs to address the following issues:

- A model of an agent which will allow interagent cooperation, and
- A model of an operational evironment which requires interagent cooperation.

In this section, we focus on the agent model while defering a model of the operational environment to the next section.

3.1 General agent architecture

Agents in our system are modelled using Deen's generalised architecture [3]. This proposed architecture is ideal for interagent cooperation because of its following characteristics:

- It provides an open interface which enables an agent to join a cooperative system easily, and
- It enables each agent to have knowledge of its operational environment. An agent will, therefore, know in advance the capabilities and skills of its acquaintances. This prevents unnecessary communication among agents and thus increases the efficiency of the system.

We shall now discuss the structure of a general agent and the various types of agents in our environment.

3.1.1 Structure of an agent

An agent is divided into two parts: *HEAD* and *BASE*. In this framework,the agents send messages between their respective heads. These messages are recognised by the head and acted upon by the base.

Head

The head is responsible for social interactions with other agents. In other words, it is like a coordinator sending and receiving messages to/from other agents. As the head is responsible for interactions, it will therefore contain:

- Data structures to describe the environment in which the agent is operating.
- A mechanism for constructing and sending messages, and interpreting messages received.
- A mechanism for detecting and resolving conflicts occuring in the environment.

The head consists of the following components:

- *Communicator*: This will act as an interface to the agent's outside world. It prepares, sends and receives messages to and from the outside world.
- *Home model*: This gives a description of the agent and its characteristics. The home model is similar to a local schema in a distributed databases.
- *Environment model*: This provides a partial model of the working environment. This model is similar to a partial global schema in distributed databases.

Base

This is like a command processor which performs tasks requested by the head and returns results to the head. It is responsible for the internal operations of an agent. We treat the base of an agent as a blackbox and therefore, concentrate on the structures and operations of the agents' head.

3.1.2 Types of agents

We can divide agents in the ATC environment into two superclasses: *Decision maker* agents and *Information server* agents. This classification allows agents belonging to the same class to share the same properties and makes the creation of members of the same class easy.

Decision maker : This superclass of agents performs a task or set of tasks. Each task is performed by making use of a number of resources. The agents reason with

the resources (which may themselves be agents) and propose a solution to a problem. The classes of agents belonging to this superclass are: *Air Traffic Controller (ATCO)* agents and *Aircraft* agent. These classes of agents have subclasses as shown in the figure 2 below.

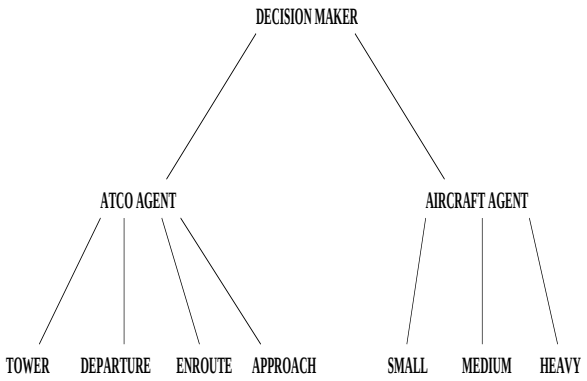

Figure 2: Taxonomy of ATC agents

Information server : This superclass of agents is used to store and supply relevant information about the ATC environment. The appropriate member for this superclass is the *Directory agent* whose contents and operations are also described later.

Detailed contents of these classes have been given in section 3.3. However, to avoid repetition, common relations and data structures shared by the agent classes are described below.

3.2 Relations and data structures

The following relations and data structures often appear in different agent classes described in section 3.3.

Agent Relation (AR) holds information on the agent's name, identity of the agent and address which identifies the machine upon which this agent is located.

```
AR[agent-name, agent-id, address]
```

Skill Relation (SR) holds information on the skills of an agent and has the following attributes.

```
SR[skill-name, skill-id, skill-description, agent-id]
```

Acquaintance Relation (ACQ) provides explicit models of other agents in the environment. This might be knowledge of the skills, locations, etc of other agents. The attributes of this relation are:

```
ACQ[agent-name, agent-id, address, skills]
```

Message Relation (MR) represents communication messages and has the following attributes.

```
MR[message-name, message-type, source, destination,
   time, message-body, replyformat, messagepriority]
```

Route map (Route): This contains much of the information representing airways together with beacons and intersections, which is useful to ATCO and aircraft. Information will be extracted from and added to the route map during evaluation of a flightpath. Figure 3 below shows an aeronautical chart data of a route map.

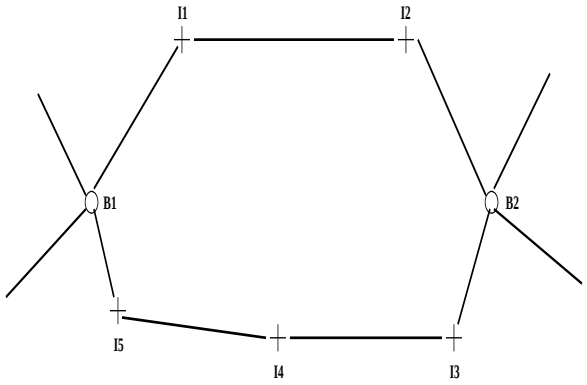

Figure 3: Aeronautical chart data

where **I** represents intersection point of airways and **B** represents beacon. This information will be retained in the form of a nested relation with the following attributes:

```
Route[AirwayName, Beacons, Intersections]
```

Flightplan (FPN): A flightplan is a complete specification of an aircraft's intentions from take-off to touch down. It is submitted as a requested route of flight, subject at all times to ATCO's modifications. The basic flightplan representation (ie, aircraft's written request), is as follows:

```
BA370  MED  150.00  1530   8000.00  TAKOFF  LHR  A10
BRAVO JUPITER A20 ENDI  HANDOFF APPRCH
```

In this example, aircraft BA370 of medium type has requested to fly at 150 knots. It will depart at 1530 from London Heathrow airport and will climb to 8000 feet while flying along A10 airway. It will fly along A10 until BRAVO intersection

then fly direct to JUPITER intersection to pick up A20 airway. The aircraft will be under enroute control until reaching ENDI beacon where it is handed off to approach control for landing.
The flightplan information will be represented in form of a relation:

```
FPN[Aircraft-id, Aircraft-type,
    EstimatedDeparTime, Requested-altitude, Route]
```

Flight Progress Strip (FPS) : This is an actual dynamic data structure which represents an aircraft's instantaneous position and vector of motion. A flight progress strip is generated from a flightplan. All modifications to the original flightplan are recorded on a flight progress strip. It also provides a rough indication of the number of aircraft an ATCO is expected to handle.
The flight progress strip will be represented in form of a relation and will have the following attributes:

```
FPS[Flightplan pointer, Instantaneous posn(x,y),
    Currentaltitude, Currentheading, Currentspeed,
    Assignedaltitude, Assignedheading, Assignedspeed]
```

The current fields represent the current motion-vector of aircraft and assigned fields represent the controller's assignments usually done for conflict resolution purposes.

3.3 Description of agent classes

As mentioned earlier, we now describe the contents of our agents using the concepts presented in the previous sections.

ATCO agent : This is the most important part of the simulation since it must be able to perform the following tasks: *monitoring*(that aircraft are on track), *planning* the trajectory of aircraft and *communicating* with other agents in the environment. To perform these tasks, an ATCO agent will have information on the aircraft's intentions from take-off to touch down, the status of aircraft which includes amount of fuel, aircraft's instantaneous position and vector position and information on its environment under its jurisdiction. This information will be retained in form of a relational database. There are five relations holding this information as shown below.

```
AR[agent-name:string; agent-id:string;
   address:string]
SR[skill-name:string; skill-id:string;
   skill-des:string; agent-id:string]
ACQ[agent-name:string; agent-id:string;
    address:string; skills:list[skillname:string]]
FPS[.................]
Route[..............]
MR[message-name:string; message-type:string;
   source:string; destination:string; time:TIME;
   messagebody:memo; replyformat:string;
   messagepriority:string]
```

The agent relation, skill relation and flight progress strip relation describe the home model of the agent. The Route map and acquaintance relations describe the environment model and the message relation is used by the communicator.

Aircraft agent : This agent represents the aircraft that exist and move within the airspace. It has the ability to interact with the ATCO agent to receive instructions and supply requested information. It also interacts with the directory agent to acquire information on the environment such as addresses of ATCO agents and current weather conditions. In order to perform its task of flying, an aircraft agent requires information on its flightplan and on its environment. The information is held in the relations shown below:

```
AR[agent-name:string; agent-id:string;
   address:string]
SR[skill-name:string; skill-id:string;
   skill-des:string; agent-id:string]
ACQ[agent-name:string; agent-id:string;
    address:string; skills:list[skillname:string]]
FPN[.....................]
MR[message-name:string; message-type:string;
   source:string; destination:string; time:TIME;
   message-body:memo; replyformat:string;
   messagepriority:string]
```

The agent relation, skill relation and flightplan relation describe the home model of an agent. The environment model is described by acquaintance relation. The message relation is used by the communicator.

Directory agent : Details on all agents in the environment are held in this agent. It will also contain information on the condition of the environment such as weather

data. This agent also describes the relationships among agents. The information is retained in the form of a relational database. The relations holding data on the state of the environment are: Agent relation (AR), Skill relation (SR) and Flightplan relation (FPN) as shown below.

```
AR[agent-name:string; agent-id:string;
   agent-type:string; address:string]
SR[skill-name:string; skill-id:string;
   skill-des:string; agent-id:string]
FPN[.............................]
```

An agent apparently gets information about its acquaintances from the directory agent. Thus the acquaintance relation held by each agent is a view of the agent relation held by the directory agent. As the flight progresses, the acquaintances change, and these changes must be replicated to each relevant agent.

Each agent has the following general operations:

1. Finding and resolving conflicts within its environment.
2. Monitoring the status of the environment.
3. Evaluating the flightpaths after resolution of conflicts.
4. Updating the agent's acquaintance list when new agents are added to the environment
5. Preparing and sending messages to other agents, and receiving messages from other agents.

4 MODEL OF AN OPERATIONAL ENVIRONMENT

As stated in subection 2.1, the ATC system is initiated by an aircraft agent requesting approval of a flightplan from an air traffic controller agent which grants permission to fly a requested route. As the aircraft agent travels along its approved flightpath, it passes from one sector to the next and hence from one controller agent's jurisdiction to that of another. During this phase, aircraft agents are subjected to variations in flightplan parameters due to conflicts. Conflicts occur mostly when there is a high density of air traffic in airspace. In this situation, two or more aircraft agents may violate the statutory separation distances. A conflict can be resolved by a controller agent changing flightplan parameter(s) of one or more aircraft agents, in consultation with other relevant controller agents.

4.1 Conflict detection and resolution

A conflict is a situation in which one or more aircraft get too close to each other. We assume that aircraft involved in the same conflict form a conflict set, in which each aircraft is in conflict with each member of the set. For example, if *S* is a conflict set consisting of aircraft *X, Y , Z* then conflicts occur between *X* and *Y*, *X* and *Z*, and *Y* and *Z* as shown in figure 4 below.

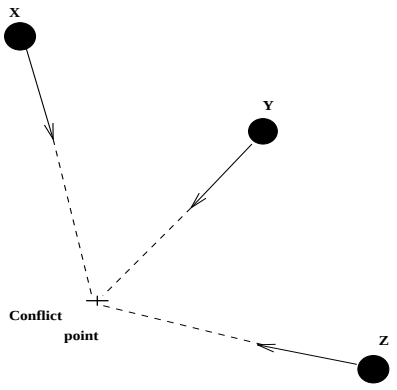

Figure 4: Conflict among aircraft agents *X, Y* and *Z*

To resolve the conflict, the ATCO agent first has to identify which aircraft (it may one or more aircraft) to change some flight parameters. To identify the aircraft, the agent will apply the following conditions:

1. Aircraft which will be in collision in the shortest time with other aircraft.
2. Status of an aircraft, which will involve amount of fuel available and emergency factors such as hijacking attempt or critically ill person.
3. Aircraft involved in a higher number of conflicts as shown figure 5 below.

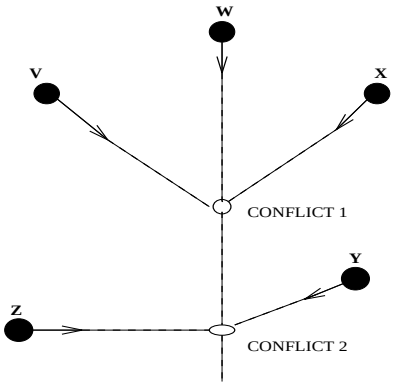

Figure 5: Aircraft agent *W* in the above scenario is involved in two conflicts.

We will use the concept of event-response as a mechanism for accomplishing conflict detection and resolution. An *event*, in our case, is a conflict detection and a *response* is a set of instructions constituting possible remedies to the conflict. We have identified the following instructions for resolving conflicts:

- Moveto *(+/-) x* feet to a free slot.
- Turn *(+/-) x* degrees
- Change *(+/-)* speed in knots
- Hold in stack circle

The above instructions constitute the type of information to be passed from ATCO to Aircraft. Information to be passed from Aircraft to ATCO is status based (e.g, amount of fuel, emergency on board, etc). Information requests occur between ATCO and Directory agents; and Aircraft and Directory agents. Similarly, information passed between ATCOs is updated data about an aircraft's flightpath. This information flow is illustrated in figure 6 below:

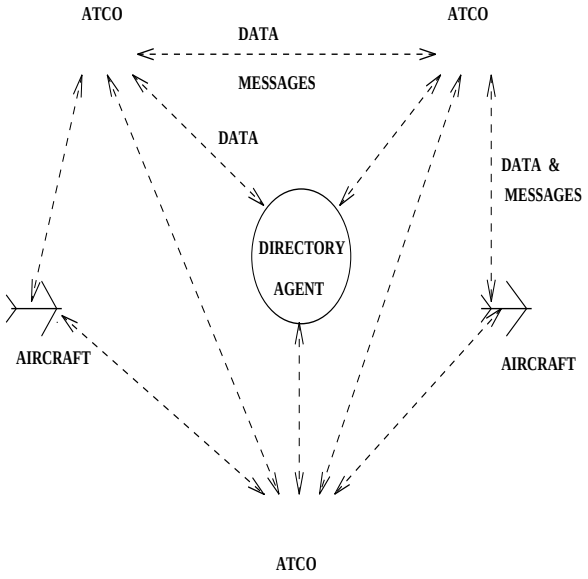

Figure 6: Cooperation of ATC agents

Thus, this information passing between agents constitute a cooperation protocol. This protocol is used to resolve various types of conflicts among aircraft agents.

4.2 Types of conflicts

In this subsection, we present a taxonomy of conflict types in ATC and abstract procedures to detect and resolve these conflicts.

1. **Separation conflict:**
   An ATCO agent's primary responsibility is to prevent mid-air collisions by ensuring minimum separation standards are met at all times between aircraft. Therefore, separation conflicts occur when a violation of these standards is found, and involves both horizontal and vertical separation of aircraft. The event-response algorithm for recognizing and solving such separation conflicts is:

   EVENT

```
FOR all aircraft in a conflict set
calculate distance between each aircraft pair
IF distance < critical distance
THEN trigger alarm and respond
```

   RESPONSE

```
FOR all aircraft
 BEGIN
  1. Find most flexible aircraft (ie candidation)
  2. Choose appropriate actions for this aircraft
        i)   Change of route of aircraft
        ii)  Change of direction
        iii) Change of flight level
        iv)  Regulation of speed
        v)   Hold aircraft in a circular state
  3. Instruct it
        i)   Moveto (+/-) x feet to free slot.
        ii)  Turn (+/-) x degrees (left/right)
        iii) Change (+/-) speed in knots
        iv)  Hold aircraft in stack circle
  4. Rollback (if unconfirmed) and ask next aircraft
 END;
```

   RESPONSE provides a choice of actions which may resolve the conflict during aircraft's flight. The ATCO agent has to select actions which will provide the most productive conflict resolution (this is illustrated later with an example). Each of the actions has a cost consequence and freedom of motion qualifier. Therefore, the best choice minimizes cost and maximizes freedom. When resolving a conflict, an aircraft is assigned an altitude different from what it requested. After the conflict situation has passed, the ATCO agent has to attempt to reassign the aircraft's original cruising altitude as follows:

```
BEGIN
 IF aircraft's assigned altitude is (+/-) x feet
  from altitude requested in its flightplan OR
  direction is (+/-) degrees from requested heading
 AND aircraft is not being handedover to another sector
 AND aircraft is not on the final phase of its flight
 AND no future conflicts will occur
 THEN  climb/descend aircraft to original altitude
       OR change heading to original heading
 ELSE do nothing.
END;
```

2. **Time**: An example of a time conflict is when an aircraft attempts to begin an approach at the same time another aircraft is already performing one. An algorithm for detecting and resolving such a conflict would be:

   EVENT

   ```
   FOR  each aircraft pair
   IF both aircraft agents are in approach mode
   THEN trigger alarm and respond
   ```

   RESPONSE

   ```
   Choose aircraft which began approach last;
    BEGIN
     Determine time remaining on approach for
       first aircraft;
     Choose appropriate action
          i)  Regulation of speed
          ii) Hold in stack circle
     Instruct aircraft agent
     END;
   ```

   As the resolution of this conflict requires a speed reduction or a hold command, the ATCO agent has, after the conflict has been resolved, to release aircraft from speed or holding restrictions.

3. **Sector Handover**: When handing over a cruising aircraft to another sector, eastbound aircraft must be at odd thousand feet altitudes(7000, 9000, etc) and westbound aircraft at even thousand feet altitudes to ensure a 1000 feet separation.

EVENT

```
FOR  each aircraft
IF aircraft is being handed over to an adjacent
   sector
AND (it is eastbound and its altitude/1000 is even
   OR it is westbound and its altitude/1000 is odd)
THEN trigger alarm and respond
```

RESPONSE

```
BEGIN
  Choose appropriate action
  Instruct it
  Rollback if unconfirmed
END;
```

To recapitulate, a conflict can have multiple responses, one of which must be selected as the most productive for any given situation. It is possible that, at any given time, resolving one conflict may also resolve one or more of the other conflicts in the same area or cause other conflicts. For example, a separation conflict between aircraft *X* and *Y*; and between aircraft *Y* and *Z*, might not require two commands if a change in the altitude of aircraft *Y* would suffice.

Since numerous conflicts may occur simultaneously, the system must pre-process these conflicts to determine which one is serious. Therefore, conflicts must be ordered based on severity and the one with a highest severity is resolved first. Any changes, made to an aircraft's flight parameters in resolving a conflict, are recorded on the Flight Progress Strip (FPS Relation) and then propagated to all ATCO agents handling the aircraft. The algorithm below represent these ideas.

```
BEGIN
 Detect conflicts among aircraft agents using EVENT;
 Select conflict with the highest severity factor;
 Find most flexible aircraft agent to change flight
 parameters;
 FOR each possible action to the chosen conflict
  BEGIN
   Determine the parameters of the instruction;
   Check suitability of instruction; (this allows
   best instruction to be identified)
  END;
 Issue best instruction;
 Update Flight progress strip relation;
 Broadcast changes in flight parameters to
 affected ATCO agents;
END;
```

4.3 Communication

Communication is vital in any CKBS environment because it is the process that allows interaction between agents. Communication between agents can happen in two ways: Communication by information sharing and communication by message passing. Communication by information sharing occurs in blackboard systems [5, 7] whereas communication by message passing is used by systems applying a common language to express their messages and exchange them according to a given protocol as in actor systems [10, 8]. Since a blackboard system is communication intensive, our system uses message passing.

Agents communicate by sending messages addressed to individual agents or to a group of agents. Each agent is identified by its unique address which is composed of an agent's name and the machine on which it is running. A message received by an agent contains the address of the sender of the message, which the receiving agent uses to reply to the sending agent. Messages are queued in order of arrival in a message buffer, but may be prioritized depending on the contents of the message. An agent interprets and responds to the messages using its own engine which is the *communicator*. Each agent can also query the directory agent for addresses of agents not in its acquaintance model.

Interaction among agents can be analysed by considering the type of messages which may be sent between different agents in the system. It should be noted that the message must be meaningful to the receiving agent (i.e, should be relevant to the receiver's knowledge). The types of messages identified for our system are: *retrieving of existing information, update of information, issuing* and *acknowledging commands of clearances.*

5 SIMULATION AND IMPLEMENTATION ISSUES

5.1 Simulation

The simulation of our model has been designed with features described in the next subsection, and it is currently being coded.

The simulation should provide responses similar to those expected in real ATC environment with appropriate agents and conditions. It is on account of the emphasis on conflict detection and resolution that our simulated ATC system is known as a Conflict Management System (CMS). CMS is a simulation of interagent cooperation among air traffic controllers, and between air traffic controllers and aircraft during the flight phase. During this phase, aircraft are subjected to variations in flightplan parameters such as time of departure, flight level, time of entry into a sector, change of course, etc.

Purpose of simulation

The objective of our simulation is to see the interaction of agents in controlled airspace. The simulation will be implemented using CKBS principles. Each agent will have a defined task (or set of tasks) to perform and will be required to communicate and co-operate with other agents. The simulation will concentrate on the handover of aircraft from one sector to another, since this requires cooperation between adjacent sectors. It will show how agents detect and resolve any flightpaths conflicts arising at this interface and secondly, the simulation will also consider the detection and resolution of conflicts within a sector as this requires cooperation between a controller agent and aircraft agent.

5.2 Simulation components and implementation

Although it would be desirable to construct a simulation of the entire ATC system, time constraints make this unfeasible. Therefore, our simulation will consist of one airport and three sectors (for *TMA* and *Enroute control*). Aircraft agents will be simulated as departing from an airport and following their respective flightplans, whilst others will enter the simulated boundary either for landing (via a waiting stack) or as enroute traffic as shown in figure 7 below.

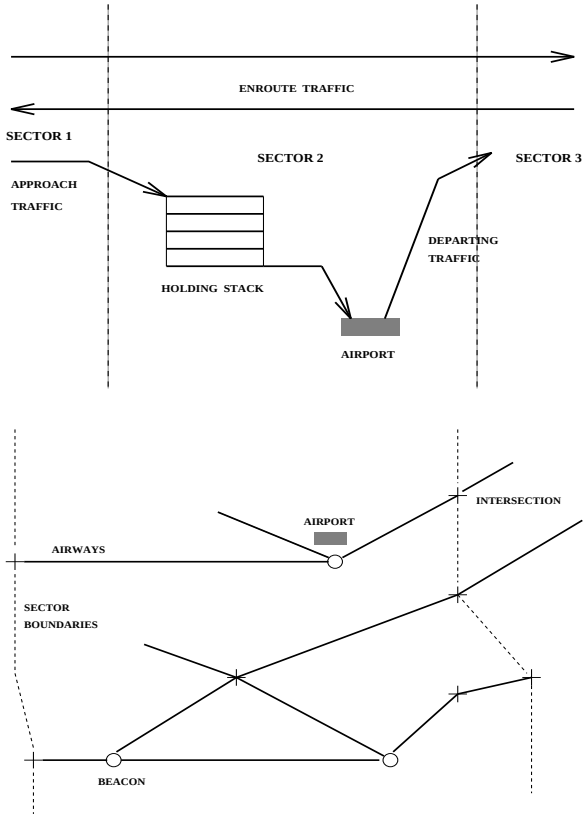

Figure 7: Elevation and Plan of simulation boundaries

An ATCO agent is the most complex part of the simulation since it must be able to perform a number of tasks such as conflict detection, conflict resolution, etc. ATCO agents will require access to a number of resources resources normally associated with air traffic control. These include flightplans and flight progress strips. Since simulation will be unable to utilize real radar data (because aircraft are simulated), it will be necessary to provide artificial radar information to the relevant ATCO agent depending on positions of aircraft agents.

For implementation purposes, we require a high level language which should be able to describe the agent heads and inter-agent activities. Due to complex data to be stored and manipulated, the language should have good data creation, modification and structuring facilities.

6 CONCLUSION

We have looked at the operation of air traffic control. This has given us a better understanding of how air traffic controllers (ie ATCO agents) cooperate in detecting and resolving conflicts occuring in mid-air. In this paper, we have presented an ATC model for inter-agent cooperation controlled airspace. The model is based on the functional classification of agents and an operational environment.

Within this paper, much emphasis has been placed on conflict detection and resolution in ATC domain. In order to perform this task, agents must cooperate with one another as they possess only local views of the environment. We have presented algorithms that can be employed by our cooperating agents in detecting and resolving conflicts. In our model, agents exchange their local knowledge with each other by message passing.

Since it is not desirable to construct a simulation of the entire ATC system due to time constraints, we are simulating and implementing a small part of an ATC system to verify our ideas.

ACKNOWLEDGEMENTS

I wish to express my thanks and appreciation to my supervisor **Prof. S. M. Deen** and **Martyn Fletcher** for their invaluable suggestions during the writing of this paper. I am also most grateful to the other members of the Data and Knowledge Engineering Centre for their valuable contributions and criticisms made during discussions.

REFERENCES

[1] Alan H. Bond and Les Gasser. An analysis of problems and research in DAI. In Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*. Morgan Kaufman Publishers, Inc., San Mateo, California, 1988.

[2] Stephanie Cammarata, David McArthur, and Randall Steeb. Strategies of Cooperation in Distributed Problem Solving. In A.H. Bond and L. Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 102–105. Morgan Kaufmann, California, 1988.

[3] S. M. Deen. Cooperating agents - a database perspective. In S. M. Deen, editor, *CKBS'90: Proc. of the International Working Conference on Cooperating Knowledge Based Systems*, pages 3–29. Springer-Verlag, London, 1991.

[4] G. Duke. *Air Traffic Control*. Ian Allan, second edition, 1986.

[5] L. D. Erman, F. Hayes-Roth, V. R. Lesser, and D. R. Reddy. The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. *Comput. Surveys*, 12:213–253, June 1980.

[6] D. Graves. *United Kingdom Air Traffic Control*. Airlife Publishing Ltd, 1989.

[7] Barbara Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26:251–321, 1985.

[8] C. E. Hewitt. Offices are open systems. *ACM Transcations on Office Information Systems*, 4:271–287, 1987.

[9] J. A. Hughes, D. Z. Shapiro, W. W. Sharrock, and R. Anderson. The automation of air traffic control. Lancaster Sociotechnics Group, October 1988.

[10] H. Lieberman. Concurrent object-oriented programming in act1. In M. H. Huhns, editor, *Distributed Artificial Intelligence*. Morgan Kaufman Publishers, Inc, Los Altos, California, 1987.

[11] D. S. McLoughlin. *Air Traffic Service Operational Overview*. Siemens Plessey Radar Limited, first edition, 1991.

[12] S. Ratcliffe. Air Traffic Control and Mid-Air Collisions. *Electronics and Communication Engineering*, pages 202–207.

[13] Randall Steeb, Stephanie Cammarata, Fredrick A. Hayes-Roth, Perry W. Thorndyke, and Robert B. Wesson. Distributed Intelligence for Air Fleet Control. In A.H. Bond and L. Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 90–101. Morgan Kaufmann, California, 1988.

[14] Perry W. Thorndyke, Dave McArthur, and Stephanie Cammarata. AUTOPILOT: A Distributed Planner for Air Fleet Control. Rand Corporation, 1987.