ACCS Winter School

ATC Simulator Project
Demonstrator Concept #3

Peter Lindsay 5 July 2005

This document describes the third in a series of highly simplified models of air traffic controller behaviour that were used in an ACCS project to help develop & evaluate an early version of the ACCS's evolving "ATC simulator framework".

The ATC task and environment given here represent only part of the complexity of real ATC tasks and environments. The purpose of the project was not to model en-route ATC in its entirety, but instead to provide requirements for the ATC simulator and to investigate the use of the *Prolog* logic programming language as a way of implementing "operator choice models".

Section 1 gives an overview of the project goals and simplifying assumptions. Section 2 defines key terms and concepts. Section 3 describes the operator task model used. Section 4 specifies the main functions used in the stochastic model, and initial settings for some of their parameters. Sections 5 & 6 specify auxiliary functions used in the model.

1. Overview

This working paper describes requirements for a third "demonstrator" project. The Demonstrator 3 concept presents a more sophisticated (though still highly simplified) model of operator behaviour than the previous two demonstrators. Methodologically, it is based on the "Operator Choice Model (OCM)" approach developed in the SafeHCI project and being used in the ATC Workload project, in which behaviours are expressed as a set of interleaved activities, together with formulae for "stochastic" data: namely, estimated duration of the chosen activity; and at choice points, the probabilities of each of the possible follow-on activities.[1] Part of the motivation for Demonstrator 3 is to establish a way of translating OCM-like models into Prolog implementations.

The main features of the Demonstrator 3 concept are the following:

- [The task & environment] As before, aircraft fly on straight line paths at fixed speeds. They may be flying level, climbing or descending when they first enter the sector. The controller's task is to resolve conflicts. Flight level change ("clearance") is the only action available to the controller.
- [The controller's behaviour] The controller exhibits reflective, adaptive behaviour:
    - The different controller activities are interleaved, and the controller "remembers" things about the different aircraft pairs, such as whether they require attention.
    - The controller considers the possible downstream effects of actions before taking them, and monitors the effects of actions after they have been taken, to ensure their effectiveness.
    - The controller adapts his/her behaviour to take account of traffic load: when traffic load gets high, the controller may sacrifice efficiency for safety when taking actions (ie, act earlier).
- [A set of basic skills] The controller's behaviour is broken down here to a set of functions representing basic skills that get used in different activities. These include:

[1] The OCM approach thus supports Monte Carlo analysis.

- a considered analysis of whether a pair is a conflict;
- a way of generating a set of strategies for resolving conflicts applicable to the current situation;
- a "watchlist" function for quickly identifying which aircraft might be affected by a given action; and
- a function for determining his/her current traffic load.

Note that the above choice of controller activities and skills, and the factors that shape operator performance of them, are the subject of ongoing investigation in the ATC Workload project. The descriptions given below are thus a "first cut" high-level attempt, and will no doubt change over time. In this Demonstrator all functions will be defined in terms of pairs of aircraft rather than individual aircraft – an assumption that is expected to make implementation simpler.

2. Terms & definitions

Terminology and more simplifying assumptions:

- A **Flight Level (FL)** is a horizontal layer of airspace, at 1000ft intervals
  - ATC convention is to indicate them in 100's of feet
    - so eg FL310 means the layer between altitudes 30,500ft and 31,500ft
  - In what follows, an aircraft is said to be at **FL X** if its current altitude is between (X*100)+/-500 feet
- A **nautical mile (NM)** is about 1.85 km
- Speeds are given in **knots**: nautical miles per hr
- **Lateral separation**: the horizontal (2D) distance between two aircraft
- **DOMS**: distance of minimum (lateral) separation
- **Separation violation (SV):** a pair has lateral separation less than 5NM while at the same flight level
- **Time to separation violation (ttsv):** the time remaining before a conflict pair will violate separation; this is the time at which they come 5NM apart, if they're both in the same flight level, or the time at which the second aircraft enters the first aircraft's flight level, if it does so at a distance of less than 5NM
- The only operator **action** we consider here is that of issuing a new **FL clearance** to a particular aircraft.
  - Scenarios can include aircraft that spontaneously appear at "arrival points" and/or disappear upon reaching "departure points" during runs of the simulator, not just at the start or the end of the run.
  - Aircraft will continue climbing or descending until their cleared FL is reached, and then they'll fly level in the centre of the FL.
  - We assume climbing & descending have no effect on horizontal speed
- **CFL**: Cleared Flight Level
- Two aircraft are in **conflict** if they will violate separation at some time in the future, unless their flight plans get changed. For the purposes of this exercise, only changes to their cleared FL will be considered.

3. The task model

Figure 1 describes the task model: ie, the set of possible activities and the flow of "control" (operator attention) between them.

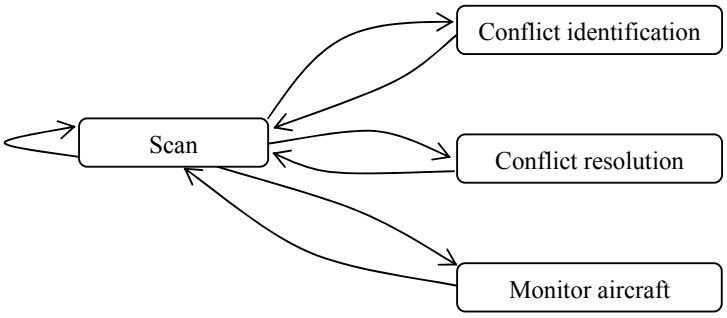

Figure 1. Task model, consisting of operator activities & control flow

Operator memory

We use a very simple model of operator memory here, in which the controller simply associates one, and only one, of the following possible "states" with each pair of aircraft: [2]

- **NotScanned**: the pair has not yet scanned (ie, controller has no previous memory of this pair, or the pair is not currently an issue but may become one in the future)
- **InConflict**: the pair is probably in conflict, and will require action
- **NoActionRequired**: the pair is probably not in conflict, and will not require action
- **NeedsMonitoring**: the pair needs monitoring because it has been recently acted on (ie, a new clearance has been issued and acknowledged) but the effectiveness of the action is not yet clear.[3]

The scanning activity

- The controller scans the screen for possible conflicts[4] and chooses one of the following follow-on activities (see Figure 1):
  - Select a pair for classification
  - Select a conflict pair to act on[5]
  - Select a pair to monitor
  - Do something else (ie, something other than the above 3 activities)
- This is a probabilistic function. It returns a probability for each of the above outcomes, for each of the pairs, representing the likelihood that the particular activity will be undertaken next.

---

[2] We'll probably add memory of individual aircraft attributes later, but this treatment sufficies for now & keeps the model simple.

[3] We may later extend this notion to support Averty's notion of Traffic Load Index (TLI), and/or the notion of watching for "level burst-throughs". In later versions, the need for sequencing will be another reason for monitoring. This part of the model is very fluid.

[4] This approach assumes the operator is continually active, and that all of their "spare time" is filled with scanning the screen. It would probably be better to replace this activity by a more general "cognitive executive" (meta-)activity, whose responsibility is simply to choose which activity to undertake next.

[5] In later versions we will probably separate planning from taking action. For now, the only actions we will consider are those which are taken immediately; later however the plan may have a time component (probably, an earliest and latest time that the action can safely be taken) eg to allow another aircraft out get out of the way.

- The function will be specified using auxiliary functions ClassifyWeight, ActionWeight, MonitorWeight and WaitWeight described below. Each of the functions returns a "weight" for the choice. The probability of a particular outcome is then the weight of that outcome divided by the sum of the weights of all possible outcomes.
- The time taken is short, and probably linear in the number of converging pairs on the screen.

The conflict identification activity

- The controller considers a particular pair of aircraft, and classifies it as InConflict, NoActionRequired or NeedsMonitoring.[6]
- The activity results in the operator's memory being overwritten with the new classification for the pair.
- The activity uses the Classification function described below.
- The time taken is that for the Classification function.

The conflict resolution activity

- Given an InConflict pair, the controller acts on one or both of the aircraft to try to resolve the conflict without introducing new conflicts.
- The activity results in revised clearances being issued to one or both aircraft. The state of the pair changes to NeedsMonitoring in memory.
- The activity uses the ActionChoiceList function described below to generate an ordered list of possible actions to resolve the conflict. The controller then works through the list one by one looking for a strategy that doesn't introduce a new conflict. He/she does this by applying the Watchlist function to the aircraft which is proposed for action, to determine which other aircraft it should be checked against; for each of these he/she uses a version of the Classification function to check whether the aircraft's new CFL is likely to result in a conflict. For the purposes of this exercise, we will assume they reject the strategy if the likelihood of conflict is greater than a certain value (a kind of risk threshold). A "quick" version of the Classification function would be used, whereby the time available to the controller is short.
- The time taken is the sum of the durations of the individual classifications, plus some amount for issuing the clearance.

The monitoring activity

- Given a NeedsMonitoring pair, the controller checks whether the action (if any) has been effective.
- There are three possible outcomes for this activity:
    - The pair are identified as still being in conflict with another
    - The aircraft have levelled off and the conflict is resolved
    - One or both of the aircraft are still climbing or descending.

  For the purpose of this exercise, the activity results in the operator's memory of the pair being overwritten with InConflict, NoActionRequired or NeedsMonitoring respectively.
- The activity is implemented by calling the Classification function, similar to the Conflict Resolution activity.
- The time taken is that for the Classification function.

[6] We may consider modifying this so the activity is cued by a single aircraft rather than a pair, and then making a series of conflict classifications on each of the aircraft returned by the Watchlist function.

The do something else activity

- For now, the operator just waits a given amount of time before returning to scanning.

4. Description of weight functions

The following functions are used in determining the probability of which activity will be undertaken next:

- ClassifyWeight, ActionWeight, MonitorWeight, WaitWeight

Each of the functions is described below in the following way:

- Name & brief description of the function and its inputs
- Its outputs
- The factors (variables) that affect the values of the function's outputs

Each function assigns a measure ("weight") that is used in the scanning activity to determine the probability that the corresponding activity is taken on the given pair at the next step.

4.1 Classification weight

- ClassifyWeight: Given a pair, determines the weight of the conflict identification ("classification") activity for that pair.
- Returns a non-negative real number.
- For now we assume that all pairs have been on the screen for some time, and that the weight is primarily determined by the operator's memory and the geometric/dynamic properties of the pair. In reality however many other things could "cue" the operator's attention, such as: the arrival of a new aircraft; a handoff request; communications from pilots; alerts. Performance shaping factors include:
    - Memory: eg if previously scanned and classified as NoActionRequired, then weight is greatly reduced
    - geometric/dynamic properties of the pair that would be easily observable (/perceivable) by the controller: eg whether the pair is currently converging; how close they are to each other (lateral separation).

Suggested implementation:

ClassifyWeight = *0* if pair diverging or greater than say 80NM apart
C_C *mem/distance*

where *mem* = *1* if NotScanned
0.001 if NoActionRequired
0 otherwise

and $C_C = 1$ [7]

4.2 Action weight

- ActionWeight: Given a pair, determines the weight of the conflict resolution ("action") activity for that pair.
- Returns a non-negative real number.
- This would be based mainly on the operator's memory of having recently classified the pair as being InConflict. Performance shaping factors might include:
    - seriousness (as measured eg by DOMS);
    - urgency for this pair (eg time to separation violation);
    - traffic load (eg bias towards acting on it early, if traffic gets heavy).

[7] The C_X parameters are introduced to make it easier to adjust the relative weights of the different activities later, when we calibrate the models.

Suggested implementation:

ActionWeight = *0* if not InConflict
0 if InConflict, ttsv =0, pair shares a flight level
C_A if InConflict, ttsv =0, pair does not share a flight level
C_A */ttsv* otherwise

where *ttsv* is the time to separation violation and $C_A = 1500$.

4.3 Monitor weight

- MonitorWeight: Given a pair, determines the weight of the monitoring activity for that pair.
- Returns a non-negative real number.
- This would be based mainly on the operator's memory of having recently taken action on one of the aircraft in the pair. Following Averty, we assume it stays medium-high until the aircraft levels off at its new cleared level, then drops as the effectiveness of the conflict resolution action becomes more apparent. Performance shaping factors will be similar to ClassifyWeight, but also taking current altitudes into account.

Suggested implementation:

MonitorWeight = *0* if not NeedsMonitoring
C_M */(distance +* C_H *Δheight)*

where $C_M = 4$ and $C_H = 0.001$.

4.4 Do nothing weight

- WaitWeight: Gives the weight of the "do something else" activity.
- Returns a non-negative real number.
- This will probably depend on traffic load.

Suggested implementation:

WaitWeight = C_W

where $C_W = 0.003$.

5. Description of skill functions

This section loosely specifies the "basic skills" used above. Section 6 gives suggested implementation details. The functions are described in the same format as above, but with addition of how long the function would normally take to carry out (its duration).

The skills used above are:

- Classification, ActionChoiceList, Watchlist, TrafficLoad.

Conflict classification function

- Classification: Given a pair, controller does a considered analysis of whether they are in conflict.
- This is a probabilistic function, returning a probability for each of the following outcomes (as used in the memory function): InConflict, NoActionRequired and NeedsMonitoring.
- Performance shaping factors might include: seriousness (as measured eg by DOMS); urgency for this pair (eg time to separation violation); time available to the controller (eg this might be short if there are other urgent activities pending); traffic load (eg bias towards calling it a conflict and acting on it early, if traffic gets heavy).
- The time taken depends on eg: complexity of the geometry (eg whether one or both are currently changing flight levels); traffic load (high traffic load makes for quick, safe decisions).

Watchlist function

- Watchlist: Given an aircraft A and a proposed action on A, the controller quickly identifies which individual aircraft need to be checked, to see if the action would put them into conflict with A.
- Returns a list of pairs to which the conflict classification function will be applied
- Performance shaping factors would be very similar to those influencing scanning for a yet-to-be-classified pair: eg those aircraft which are converging with A and which are close to A. The suggestion is to implement this by applying the ClassifyWeight function to all pairs involving A, and selecting those that return a value above a certain threshold.
- The time taken would be equal to the number of aircraft on the screen times the time taken to apply ClassifyWeight.

Resolution strategy choice function

- ActionChoiceList: Given a conflict pair, controller formulates a set (ordered list) of possible resolution strategies.
- Here, each strategy consists of one or two actions (usually one will suffice), consisting of an instruction to a given aircraft to change its FL to a given value.
- Controllers apparently use many different strategies, and have their own personal preferences. Here the options are constrained because of the limitation that control is done via clearances alone.
- The time taken depends on eg: complexity of the geometry.

Traffic load function

- TrafficLoad: The controller is cognisant of his/her current traffic load. Part of their observed coping behaviour is to keep their traffic load from getting too high, so they have sufficient time to deal with eventualities should they arise. In periods where high traffic load is anticipated, the controller acts earlier, possibly at the cost of efficiency (eg taking action when no action was in fact required).
- The ouput is a traffic load rating (TBD) which is used in functions such as Classification above.
- Will probably follow Averty's Traffic Load Index (TLI) approach. The shaping factors thus include: number of aircraft; number of deferred conflict classifications; seriousness and urgency of pairs awaiting action; number of pairs being monitored to ensure that action was effective.
- Will probably involve functions such as ClassifyWeight
- Time involved is short (& constant?)

6. Suggested implementations

ActionChoiceList

See Figure 2**Error! Reference source not found.** for one possible heuristic for resolving conflicts by flight level change only. The strategy is based on trying to minimise the divergence from the aircrafts' original CFL. Roughly, it levels out the climbing aircraft at the level below that at which separation violation would occur. For example, if A is flying level at FL x and B intends to climb through FL *x*, but this puts them in conflict, then B's clearance is changed to FL *x-1* instead. In the figure, "(A:x)" means the CFL of aircraft A gets changed to x; the usual Prolog convention is used for ordered lists.[8]

[8] Some cases have been omitted from the analysis: eg the case where B is already at the same FL as A but trying to climb or descend. We assume action will have been taken long before such a situation arises.

Case A *of*
Flying level at FL x →
Case B *of*
Flying level at FL x →
[(A:x+1),(B:x+1),(A:x−1),(B:x−1),(A:x+2),(B:x+2),…]
At FL y, climbing to FL z, with y<x≤z →
[(B:x−1),(B:x−2),…,(B:y)]
At FL y, descending to FL z, with z<x<y →
[(B:x−1),(B:x−2),…,(B:y)]
end
At FL x, climbing to FL w →
Case B *of*
At FL y, climbing to FL z, with x<y (or x=y & w≤z) and SV at FL u →
[(A:u−1),(A:u−2),…,(A:x)]
At FL y, descending to FL z, with SV at FL u →
[(A:u;B:u+1)]
end
At FL x, descending to FL w →
Case B *of*
At FL y, descending →
Analogous to the climb-climb case above
end
end
(The missing cases are symmetric.)

Figure 2. Resolution strategy choice function for aircraft pair (A,B)