Modelling Erroneous Operator Behaviours for an Air-Traffic Control Task

Peter Lindsay **Simon Connelly**

Software Verification Research Centre,
The University of Queensland,
Brisbane, Qld 4072, Australia.
{peter.lindsay, simonc}@svrc.uq.edu.au

Abstract

This paper introduces a new approach to formalising analysis of human errors in human-computer interaction. The approach takes account of the cognitive processes involved in a task, and how mistakes arise and how errors propagate through the task. It argues for modelling errors as behaviours rather than as events (the usual approach), at least for tasks involving highly interleaved concurrent, ongoing activities. The models are formalised using a combination of CSP and temporal logic, and the approach is illustrated on a case study from Air Traffic Control. By providing a richer modelling framework and being more expressive, the approach overcomes significant limitations of existing human-error identification techniques.

Keywords: human-computer interaction, human error identification, user interfaces

1 Introduction

1.1 Motivation

Human Reliability Assessment (HRA) received much attention in the 1970's and 80's, especially in the nuclear industry following the Three Mile Island incident, and a number of different HRA techniques have emerged [Swain and Guttmann, 1983, Kirwan, 1990]. For the most part however, safety assessment of User Interface (UI) designs and interactive-system operator performance is still largely done in an *ad hoc* manner [Leveson, 1995].

Two of the main problems with applying existing HRA techniques to interactive systems is that the techniques lack objectivity [Svenson, 1989] and typically only apply to tasks involving well-defined sequence of essentially independent activities: i.e., sequential tasks in which the failure of one activity is assumed to be largely independent of the others [Kirwan, 1992]. Formal modelling has been proposed as a way of making parts of the analysis more objective, as well as providing the basis for new analysis techniques [Palanque *et al.*, 1997, Johnson, 1997, Leveson *et al*, 1997]. The second, task-modelling problem is harder to overcome and requires the development of new modelling and analysis techniques. This is particularly true for tasks involving highly interleaved, concurrent, ongoing activities – such as are involved in Air Traffic Control (ATC) for example.

A previous paper [Leadbetter *et al.*, 2001] described an ATC case study and introduced an approach to predicting the sources of possible operator errors based on formal models of the ATC system and its user interface, and of the cognitive processes involved. The formal models provided the basis for a systematic exploration of the different sources of error that could arise, using a method adapted from Hazard and Operability Studies (HAZOP) and Functional Failure Analysis.

The purpose of the present paper is to step back from looking at failures of individual activities, and instead to consider failure of the task as a whole. In particular, the paper proposes a means for formally modelling operator behaviours and propagation of errors between activities, as a way of overcoming the independence limitation of existing HRA techniques noted above.

The proposed new HRA approach involves the use of a number of interrelated models:

- cognitive models to describe how the operator's attention flows between different activities;
- event-trace models to capture activity-level failures;
- temporal logic to express the behaviours that constitute task-level failures; and
- stochastic extensions of the event-trace models to capture quantitative aspects.

The present paper describes the error identification and classification part of the overall HRA process only, and omits stochastic modelling for now. The approach is illustrated below on a simple Air Traffic Control (ATC) case study involving en-route control through a simulated airspace.

1.2 Overview of this paper

Section 2 describes the ATC simulator and task on which the error identification approach will be illustrated. Section 3 briefly describes the cognitive processes and operator actions involved in the ATC task. Section 4 introduces the model of operator choices (decisions and actions) and their system-level effects, which will serve as the basis for task-level failure analysis. The *Operator Choice Model (OCM)*, as it is called here, is formalised in Hoare's CSP notation.

Section 5 introduces temporal logic notation to describe operator behaviours as sets of possible traces through the OCM. Section 6 describes part of a top-down failure analysis based on the OCM and using temporal logic to aid reasoning. One particular task-level failure mode – failure to take action – is decomposed into the different possible behaviour types that could lead to the failure. This illustrates one of the new approach's main contributions: the use of formal models to help break a complex problem down into more manageable problems.

The description of how the approach can be extended to support (semi-)quantitative Human Reliability Assessment is the subject of a future paper. The present paper focusses on the conceptual approach to description and classification of task failure types, which is essentially new.

1.3 Related work

We briefly survey the literature on human-error modelling and formal models of ATC tasks.

As Kirwan [Kirwan, 1990, Kirwan, 1992] notes, few practical techniques have been developed for human-error classification for use in risk assessment. Human-error identification techniques mostly focus on errors in individual activities (such as Reason's "slips" and "mistakes" [Reason, 1987]) or treat them as single events (e.g. error of omission, error of commission, extraneous error [Swain and Guttmann, 1983]).

One of the best-known HRA techniques is the Technique for Human Error Rate Prediction (THERP) [Kirwan, 1990], which uses event trees with recovery paths to analyse task failures. THERP is typically used in highly proceduralised situations in which the task is broken down into a sequence of individual steps, some of which involve checks on the outcomes of previous steps. For each step, the probabilities of success, failure and recovery (if applicable) are estimated. The probability of task failure can then be calculated by combining the probabilities of errors in individual steps in branches without recovery paths. THERP is one of the few techniques which takes dependency between actions(/errors) into account quantitatively.

As noted in the ATC literature however, many tasks are more complicated than can be modelled using event trees, since they involve repetition and interleaving of events, and activities are often not performed in a predetermined order [Redding and Seamster, 1996, Seamster *et al.*, 1997]. In particular, none of the methods that Kirwan discusses apply very well to our ATC simulator case study.

A number of formal (and semi-formal) modelling techniques have however been applied to Air Traffic Control. The Eurocontrol organisation has developed a very sophisticated model of the cognitive processes involved in en-route control [Kallus *et al.*, 1999]; to the best of our knowledge it has not yet been used to analyse human error in any depth. Palanque *et al* applied Petri Nets to modelling the effect on an ATC task of the introduction of a new UI technology (data link) [Palanque *et al.*, 1997]. Johnson illustrates the use of formal models to support the findings of accident investigations, and illustrates the approach on an aircraft accident [Johnson, 1997].

Leveson *et al* document the outcomes of a system safety program for development of an ATC subsystem (CTAS) for arrival control [Leveson *et al*, 1997]. The program included modelling and safety analysis of part of the operator's "normal" behaviour. Formal models were used to analyse the possibility of "mode confusion" resulting from error-prone automation features, and to make recommendations concerning UI design. An informal human-factors analysis was also conducted, to predict the effect of introduction of CTAS on controller performance.

2 The ATC case study

The ATC case study was designed by the Australian Key Centre for Human Factors and Applied Cognitive Psychology (KCHFAPC) in consultation with ATC domain experts. It concerns a highly simplified ATC task performed on a simulator. A joint SVRC/KCHFAPC project is using the case study to develop a semi-quantitative approach to predicting risk associated with operator error. Although the task is highly simplified from an ATC viewpoint, it is sufficiently complicated to challenge existing techniques.

This section briefly describes the ATC simulator and task: for more details see [Leadbetter *et al.*, 2000]. The User Interface is described in detail for the interest of the current audience.

2.1 Overview

In broad terms, the simulator presents a highly simplified ATC system in which aircraft fly along straight-line segments – called *flight paths* – between *waypoints* within a fixed sector of airspace (see Figure 1). The primary task of the operator is to ensure that the aircraft moving through the sector remain separated by no less than the defined minimum sepa-

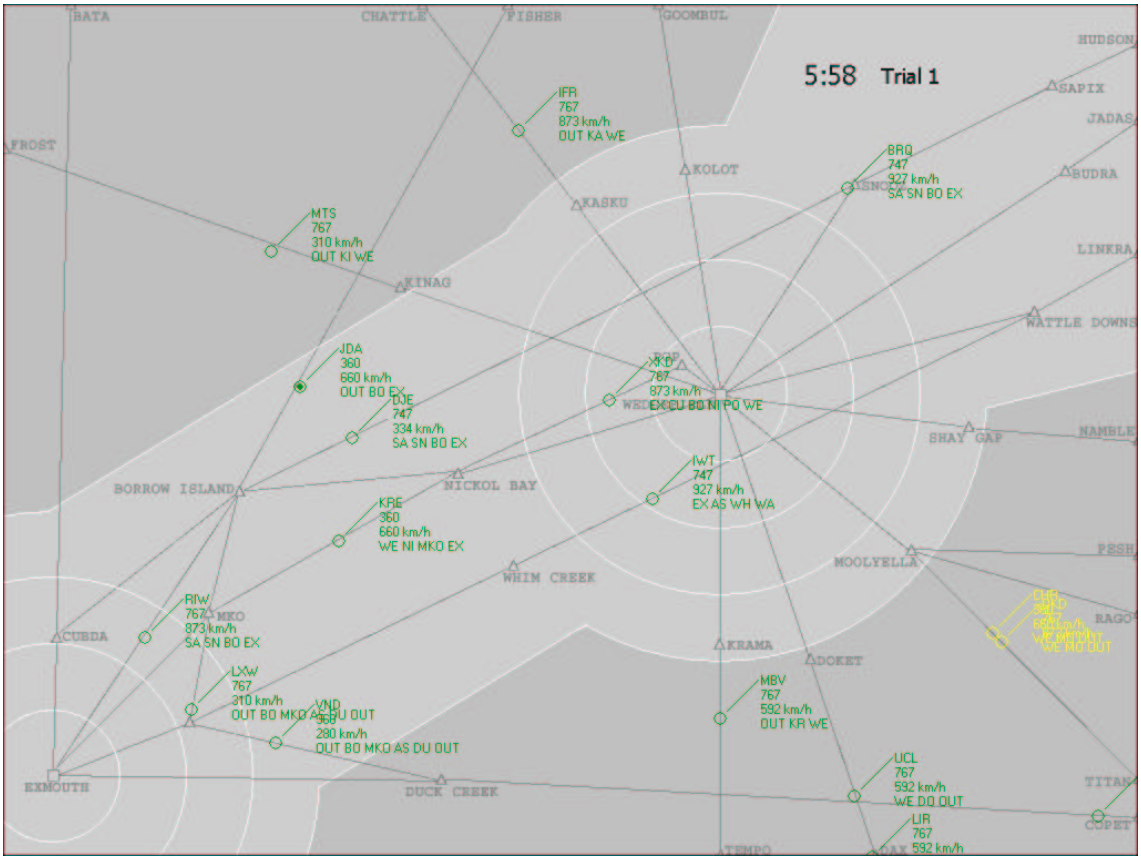

Figure 1: A (black and white) screenshot of the ATC simulator.

ration distance (5000 metres): failure of this requirement is called *separation violation*. For the simulator, the only control that the operator can exert is to change the speed of individual aircraft (see below). To keep them motivated, experiment subjects receive scores based on how fast they move aircraft through the sector without a separation violation occurring.

2.2 ATC simulator display

The ATC simulator has a display which depicts a simulated sector of airspace – consisting of airports, waypoints and flight paths – together with the location and details of aircraft currently flying within the sector: see Figure 1. Flight paths are shown as dark lines, aircraft as circles, airports as squares, and waypoints as triangles. Details of each aircraft (call sign, aircraft type, speed and flight route) are shown near the aircraft. *Flight routes* are represented as sequences of waypoint/airport codes. The display is updated at short intervals to give the impression that the aircraft are moving.

2.3 User Interface functionality

Operators interact with the simulator via two main UI functions:

- selecting (by highlighting) a single aircraft, and
- changing the speed of the selected aircraft.

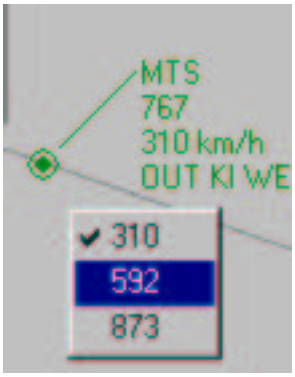

Figure 2: A partial screenshot of the Speed Menu for the selected aircraft.

An aircraft is selected by clicking the left button when the cursor is positioned over an aircraft. The selected aircraft is highlighted using a solid dot within the circle that represents the aircraft (see Figure 2). Only one aircraft can be selected at a time: when a new aircraft is selected the previously selected aircraft loses its highlighting.

Changing the speed of the selected aircraft involves three steps:

1. Opening the speed menu by clicking the right button. The aircraft must be selected before the speed menu can be accessed. The menu appears at the position of the cursor.

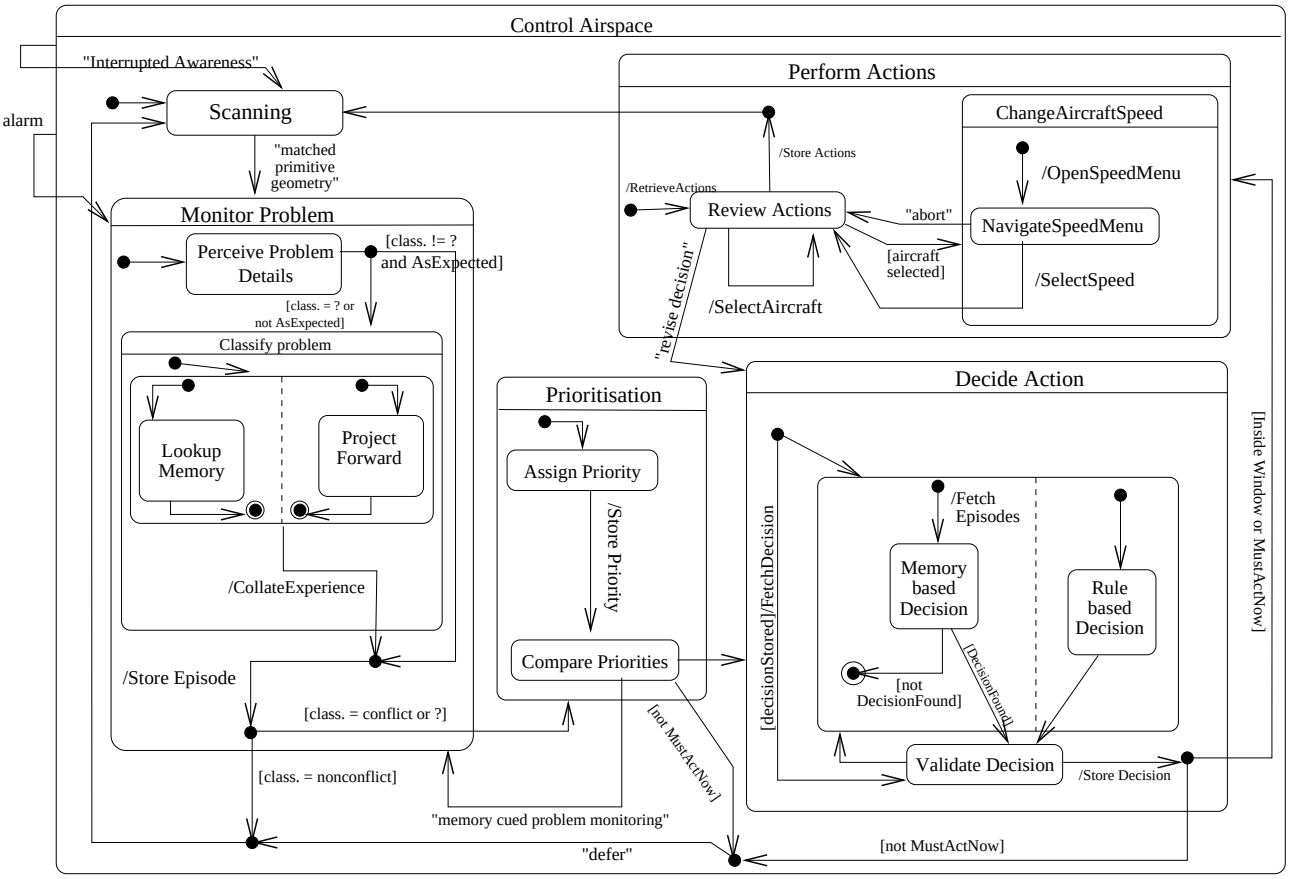

Figure 3: Control-flow model of the ATC-simulator operator's cognitive processes.

2. Navigating the speed menu by moving amongst the menu entries. The speed choices in the menu depend on the type of aircraft selected. A tick (✓) within the menu indicates the aircraft's current speed (see Figure 2).

3. Selecting a speed by left clicking on the desired menu entry.

The operator may abort this operation by clicking the left button when the cursor is positioned outside the speed menu.

3 Cognitive Model

This section outlines a *cognitive model* for the ATC case study which was developed in collaboration with Andrew Neal and Mike Humphreys at KCHFAPC. The model identifies the main cognitive processes involved and how "control" (roughly, the operator's attention) flows from one process to another under different conditions. It is included here to provide the "semantics" of the Operator Choice Model described later in the paper. As will become evident, the full cognitive model illustrated in Figure 3 is more detailed than strictly required for the purposes of this paper; its description here is thus kept brief. (The interested reader is referred to [Connelly *et al.*, 2001] for more details.)

3.1 Overview and terminology

First, some terminology: A *conflict* is defined as two aircraft being on courses that – if their speeds are left unchanged – will lead them to violate the minimum separation distance at some time while they are within the sector. For simplicity of explanation, a *problem* is defined to be a pair of aircraft to which the controller pays attention as possibly being in conflict (although in general an en-route control problem may involve two or more aircraft).

In brief, the main cognitive processes in Fig. 3 are:

- "scanning", in which the operator systematically scans the display searching for possible conflicts;
- focussing on a particular problem (pair) and classifying the pair as being in conflict or not;
- if the monitored pair is in conflict, determining an action plan for the pair, and
- performing corrective actions on the pair (i.e., setting new speeds for one or both of the aircraft via the UI).

The model is reasonably generic but is primarily intended to represent the behaviour of relatively experienced air-traffic controllers. It has been reviewed by ATC domain experts, and it is broadly consistent with the Eurocontrol task analysis of en-route control

[Kallus *et al.*, 1999] (although of course our simulator and task are far simpler than in a real ATC system).

3.2 Modelling notation and more detail

The model uses the UML statechart notation [Oestereich, 1999]. Each statechart state represents an abstract cognitive process (such as monitoring a conflict), which in some cases is further subdivided into subprocesses. The model purposefully blurs the distinction between cognitive processes and physical actions such as navigating a speed menu. For the purposes of this paper, transition decisions are made nondeterministically.

The underlying memory structures for the cognitive model are explained in detail in [Connelly *et al.*, 2001]. Briefly, the operator's long-term memory stores episode records, each of which consists of a collection of data relations representing the problem's details at different points in time. The operator's short-term memory is modelled as a small array of truncated data relations, consisting of the location and priority information pertaining to recently considered problems.

The processes in Figure 3 access memory in various ways: e.g. MemorybasedDecision in DecideAction extracts possible solutions from the collection of data relations in long-term memory; this is termed a "conflict resolution library" in [Kallus *et al.*, 1999].

4 Operator Choice Model (OCM)

This section introduces the key model for use in error identification and classification; we call it the *Operator Choice Model (OCM)*. The OCM describes the order in which certain events can take place during runs of the simulator.

4.1 Overview of approach

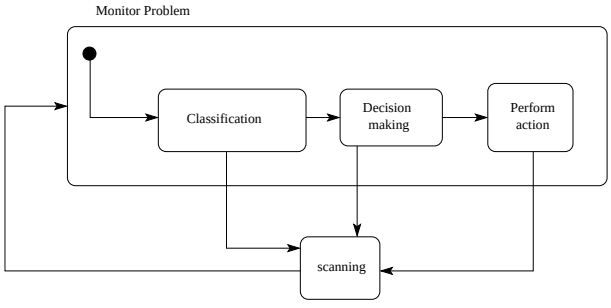

Figure 4: Operator Choice Model process

The OCM encompasses both correct and "incorrect" decisions, in order to record how mistakes arise, propagate and get corrected ("recovered from") in the course of a run of the simulator. The OCM was derived from the Cognitive Model of Section 3 by focussing on certain key events: namely, decisions and actions that critically influence the subsequent run of the simulator or later decisions of the operator. To keep analysis tractable, the set of analysed events has been kept small, and certain less commonly occurring transitions have been ignored. The events chosen for inclusion are ones that are observable (measurable) in experiments.

As a result, the OCM is modelled at a higher level of abstraction than the Cognitive Model, and the process diagram can be simplified considerably: see Fig. 4. The omitted transitions for the most part concern "jumps in control" cued by memory or alerts in the UI, such as suddenly remembering there is a higher priority problem requiring urgent attention.

As well as being more coarse-grained than the Cognitive Model, the OCM is somehow more generic – in that it describes the general logic of "handling an ATC problem". For example the OCM could be used to model the sequencing of aircraft for arrival control. (In fact, the model in Fig. 4 is really quite general and could represent almost any decision/action process. The tailoring of the model to the case study occurs in the CSP details below.)

The rest of this section is structured as follows: Section 4.2 introduces the formal notation used in defining the OCM. Sections 4.3–4.6 build up the model incrementally by describing each of the individual activities and their possible outcomes (in terms of the scenario unfolding on the simulator). The model is nondeterministic in the sense that the operator has the power to make different choices at different points. Section 4.7 constrains the model further, to reflect the fact that the simulator behaves totally deterministically.

4.2 Notation

The OCM is defined in a formal notation based upon Hoare's CSP notation for describing Communicating Sequential Processes [Hoare, 1985]. Only the part of the CSP notation dealing with (CSP) *events* and *processes* is used here.

The CSP meaning of process is close enough to our meaning of the cognitive processes being performed by the operator as to not need further explanation. The processes modelled in the OCM correspond to the main cognitive and physical processes involved in the ATC task, as outlined above. In fact, the operator part of the OCM does not use concurrency at all: at the level of granularity modelled here, the operator is considered to be only thinking about one thing at a time.

CSP events correspond to observations of the system at selected points in the above processes. They are labelled with information pertaining to the operator's intentions and the true simulator "state". The event labels are explained in more detail as they are introduced below. Generally speaking, the events "occur"

when the operator makes transitions between the corresponding cognitive processes.

CSP	meaning
$X \Box Y$	nondeterministic choice between processes X and Y
$X \parallel Y$	parallel combination of processes X and Y
$a \rightarrow X$	event a occurs then process X starts

Table 1: Some CSP notation for processes

CSP processes are defined by describing the events and transitions (to other processes) that can take place. They are described in terms of the corresponding processes in the Cognitive Model. Table 1 summarises the CSP notation used below. Note that $\Box_{i:1..n} X_i$ is used as shorthand to represent a choice between processes $X_1, \ldots, X_n$.

A *trace* of the model is the sequence of events corresponding to the choices made by the operator in a particular run of the simulator.

Pairs will stand for the set of all possible pairs of aircraft within the sector over the course of the simulation. (Not all pairs will be observable by the operator at any one time.) In what follows, certain events and processes will be labelled with subscript p, meaning they are specific to pair p from *Pairs*.

4.3 Scanning

S stands for the scanning process. It is modelled as a period of scanning of the screen followed by nondeterministic (operator) choice of a pair p to monitor as possibly being in conflict.[1] In CSP this is written as:

$$S = s \rightarrow \Box_{p:Pairs}(s_p \rightarrow C_p)$$

where s is the label for the "start of scanning" (or equivalently "end of attending to a particular pair") event, and s_p is the label for the "beginning to monitor pair p" event (i.e., the MonitorProblem state in Figure 3). Here p represents the pair the operator is thinking about, not necessarily the pair they are observing (e.g. in the case of mistaken identity).

4.4 Classify pair

C_p stands for the "classify" process for a particular pair p of aircraft; there is one such process for each p in *Pairs*. The operator tries to classify p as being a conflict or non-conflict[2] and "the environment" (the simulator) determines whether it actually is a conflict. (The model could be said to be omniscient, in that it knows more about what is really going on than the operator.) This process also includes the prioritise sub-process.

label	operator thinks	actual situation
c^1	conflict	conflict
c^2	conflict	non-conflict
c^3	non-conflict	conflict
c^4	non-conflict	non-conflict

Table 2: 'Classify' event labels

Table 2 explains the event labels. Event c_p^i represents the operator's classification of pair p, and occurs when the operator stores the episode in memory (i.e., the StoreEpisode action in Figure 3).

$$C_p = (\Box_{i:1..4} c_p^i \rightarrow S) \Box (\Box_{i:1..2} c_p^i \rightarrow D_p)$$

After classifying the pair, the operator can nondeterministically return to scanning or proceed to deciding on corrective actions (the D_p process described below). For example, even if the problem is classified as a conflict the operator may decide that its priority is too low for it to require immediate attention. Note that the operator will proceed to D_p only if they think the pair is in conflict (i.e., $i = 1, 2$). On the other hand, they can defer deciding an action and return to scanning no matter what the outcome of the classify process (i.e., $i = 1, 2, 3, 4$).

4.5 Decide action

D_p stands for the course of the "decide action" process for a particular pair p; there is one such process for each p in *Pairs*. Again, the model combines nondeterministic operator choice with deterministic simulator outcomes. Table 3 explains the labels. The event d_p^i occurs at the moment when the operator finalises their decision on how to handle the conflict (i.e., the StoreDecision action in the Cognitive Model).

$$D_p = (\Box_{i:1..3} d_p^i \rightarrow (A_p \Box S)) \Box (\Box_{i:3..4} c_p^i \rightarrow S)$$

label	operator's plan vs reality
d^1	decide on an action, and the action would resolve the conflict in the pair
d^2	decide on an action, but the action would not resolve the conflict
d^3	decide on an action, but no action is actually necessary since the pair are not in conflict

Table 3: 'Decision' event labels

After an action plan is decided, the operator can go immediately on to performing actions (the A_p process below) or return to scanning. Another possibility is

[1] At any one time only a subset of *Pairs* will actually be visible to the operator; the environment is in control of which aircraft are visible – see Section 4.7 below.

[2] For simplicity of modelling the decision is treated as a binary choice, although in practice the classification would have some kind of confidence weighting attached.

that projecting forward in DecideAction (ValidateDecision) results in the pair being reclassified as a non-conflict (event c_p^3 or c_p^4), and the operator returning to scanning.

4.6 Perform action

A_p stands for the "perform actions on pair p" process.

$$A_p = (\Box_{i:1..3} a_p^i \rightarrow S) \Box (\Box_{q:Pairs} a_q^4 \rightarrow S)$$

label	effect of action on the given pair
a^1	action resolves conflict & no new conflicts created
a^2	action does not resolve conflict, but no new conflicts created
a^3	no action was necessary for this pair (because nonconflict) but is benign although non-optimal
a^4	action turns nonconflict into conflict

Table 4: 'Action' event labels

Table 4 describes the different possible outcomes of this process in terms of the effects of the operator's actions on the selected pair. The table includes an event a_q^4 that represents the (presumably unintentional) act of *creating* a new conflict pair q as a result of modifying the speed of one of the aircaft in p and bringing it into conflict with a third aircraft.[3]

That completes the description of the choices and actions available to the operator in the general case. The rest of this section focuses on the simulator case study.

4.7 Simulator model

The CSP model above is quite general and describes all possible choices and outcomes, including some which are not physically possible. For example, since the simulator behaves totally deterministically, it is not possible an aircraft pair's conflict state to change without an explicit action by the operator. More precisely, for the purposes of the case study we can assume:

- At any time a pair p can be in conflict or not.
- The conflict status of p can change as the result – and only as the result – of the operator performing an action on one or the other of the aircraft in p.[4]

[3]Other outcomes are also possible, such as creating or resolving more than one conflict. We have developed a more sophisticated model that uses a mapping from pairs to outcomes to handle such cases, but the simpler model is given here for ease of explanation.

[4]The aircraft will be considered still to be in conflict if they have already violated separation and are moving apart.

The determination of whether or not two aircraft are in conflict assumes they do not change their behaviour in the meantime. This assumption is valid for the simulator but would not be true for real ATC systems, where e.g. the aircraft could change speed independently of the controller's command.[5]

This section adds constraints to the processes defined above, to rule out physically impossible behaviours. Process I_p below will represent the conflict states that pair p passes through, for a pair that starts life as a conflict (i.e., has been programmed into the simulator to violate separation if the operator makes no change to either aircraft's speed). N_p is the analogous process for pair which starts as a nonconflict. (I can be read as meaning "in conflict" and N as meaning "not in conflict".)

The simulator's behaviour with respect to events from the CSP model can be described as the parallel combination of processes I_p, for each pair p which begins as a conflict, and processes N_p, for each pair p which begins as a nonconflict, where:

$$I_p = (\Box_{e:\mathcal{A}_p^{cc}} e \rightarrow I_p) \Box (\Box_{e:\mathcal{A}_p^{cn}} e \rightarrow N_p)$$
$$N_p = (\Box_{e:\mathcal{A}_p^{nn}} e \rightarrow N_p) \Box (\Box_{e:\mathcal{A}_p^{nc}} e \rightarrow I_p)$$

Here $\mathcal{A}_p^{cc}$ is the set of events which are possible for a conflict pair p but which leave it as a conflict pair, and so on (cc means from conflict to conflict, nc from non-conflict to conflict, etc):

$$\mathcal{A}_p^{cn} = \{a_p^1\}$$
$$\mathcal{A}_p^{cc} = \{c_p^1, c_p^3, d_p^1, d_p^2, a_p^2\}$$
$$\mathcal{A}_p^{nn} = \{c_p^2, c_p^4, d_p^3, a_p^3\}$$
$$\mathcal{A}_p^{nc} = \{a_p^4\}$$

Thus for example, if pair p is in conflict (I_p) and the operator erroneously classifies them as not being in conflict (c_p^3), the pair none-the-less remains in conflict (I_p).

Finally, the *Operator Choice Model* is the parallel combination of the operator choice model from Section 4 and the simulator model above:

$$OCM = S \parallel (\parallel_{p:Init_I} I_p) \parallel (\parallel_{p:Init_N} N_p)$$

where $Init_I$ and $Init_N$ are the set of conflict pairs and non-conflict pairs when the simulation begins, respectively.

Thus for example, the operator is "physically constrained" by the simulator from choosing the c_p^1 event for a pair p which is not at that moment in conflict.

5 Modelling erroneous behaviours

This section introduces temporal logic as a means for describing erroneous behaviours as sets of traces over the Operator Choice Model.

[5]The modelling approach could be extended to capture non-deterministic behaviour of the "environment" (including e.g. failure to obey the ATC operator's command) if desired.

5.1 What constitutes a failure?

In order to assess operator reliability it is necessary to define what constitutes a task failure. For a task like ATC, however, it is difficult to say exactly when a failure has occurred. Separation violation is an unambiguous indication that a failure has occurred sometime during the task, but to analyse how it comes about it is necessary to study possible causes further back in the ATC process – such as having failed ever to monitor the pair during scanning, or being unable to access the speed menu for the given aircraft. Similarly, classifying a conflict pair as "probably not in conflict" may be an innocent mistake when the pair are far apart, since in the normal course of events there would be ample opportunity to reclassify the pair as they get closer and to intervene; the mistake is clearly critical however if it continues to be repeated until it is too late to take corrective action.

As a second point, although the main task-level failure of interest in the ATC case study is separation violation, unnecessarily slowing down an aircraft – or slowing it down too much – can also be considered task failures in a sense (or at least suboptimal performance). The techniques discussed below apply equally well to such cases.

Clearly it is not sufficient to consider individual errors in isolation, as done in the OCM above: a more general way of expressing erroneous behaviours is needed. This section introduces (discrete-time) *temporal logic* notation (e.g. [Ben-Ari *et al.*, 1983]) to describe operator behaviours – and their system-level effects – as sets of event traces over the OCM.

5.2 Temporal logic notation

Temporal logic extends standard predicate logic with notation for talking about the order in which events occur in time. Table 5 explains the basic temporal logic notation used in this paper. Note that the truth of a temporal formula is interpreted at a given reference point in time ("now") which needs to be supplied: see the examples below. The successive ("next") points in time correspond to the events in an OCM trace for a run of the simulator. Time "finishes" when the simulator terminates or a separation violation occurs, whichever comes first.

A single temporal formula defines a *set* of possible traces of the OCM. Temporal logic thus provides a simple and concise notation for reasoning about behaviours that abstracts away from detail about precisely when events occur.

The abbreviation

$$F.G == F \wedge \bigcirc G$$

will be used below to mean that F is true now and G is true immediately after the next transition.

temporal formula	meaning
e (event label)	event e occurs now
$\neg F$	not F
$F \vee G$	F or G
$F \wedge G$	F and G
$\bigcirc F$	F is true after the next transition
$\Diamond F$	F is true now or at some time in the future
$\Box F$	F is true now and hereafter

Table 5: Some temporal operators

5.3 Examples

This section formalises the description of some operator behaviours over the course of a simulator run. The descriptions should be interpreted as starting from some given point in time.

1. Conflict p is persistently classified as a nonconflict, every time it is noticed henceforth (if ever):

$$\neg \Diamond c_p^1 \text{ (or equivalently } \Box \neg\ c_p^1)$$

2. Pair p is not monitored henceforth (e.g. it is forgotten or ignored altogether):

$$\neg \Diamond s_p \text{ (or equivalently } \Box \neg\ s_p)$$

3. Conflict p is eventually handled correctly, without creating any new conflicts:

$$\Diamond (c_p^1.d_p^1.a_p^1)$$

4. Nonconflict p is handled as a conflict, with the result that it actually becomes a conflict:

$$c_p^2.d_p^3.a_p^4$$

6 Analysing task failure

This section illustrates how the OCM can be used to analyse and classify task failures with the aid of temporal logic.

In what follows, the temporal formulae should be interpreted as starting from the beginning of the run of the simulator.

6.1 Top-level decomposition

As noted above, the main task failure for analysis is occurrence of a separation violation, for pair p say. This can initially be decomposed into different kinds of failure by exploiting the structure of the simulator model defined in Section 4.7; namely:

- Conflict not resolved: conflict pair p was never resolved (i.e., no $\mathcal{A}_p^{cn}$ event occurs)

$$never\_resolved_p == \neg \Diamond a_p^1$$

- Conflict created: some operator action caused p to become a conflict (an $\mathcal{A}_p^{nc}$ event occurs)

  $\Diamond a_p^4$

Henceforth we focus on the "never resolved" case, and assume p is a conflict pair. Considering paths through Figure 4, there are two cases to consider:

- Non-response: no action was ever taken to resolve p (i.e. operator never makes transition to Perform Action)

  $non\_response_p == \neg \Diamond(a_p^1 \vee a_p^2)$

- Ineffective response: action was taken on p but it was ineffective, and no subsequent action was taken on p

  $\Diamond(a_p^2 \wedge non\_response_p)$

(Note that selecting the aircraft's current speed from the menu falls into the "ineffective" category, by the definitions in Table 4.)

The rest of this section analyses the "non-response" task failure for a given conflict pair p. Analysis proceeds by decomposing the behaviour type by considering Figure 4 and the "cycles" that avoid the "perform action on p" event. The diagram is useful for identifying different possible "non-action cycles", but the formal OCM model is needed for distinguishing exactly what event sequences are possible, and the temporal logic is needed to distinguish exactly which trace-sets belong to the different behaviour types.

For each failure type we briefly discuss possible causes and contributing factors, to aid understanding of the formalisation. Note however that the analysis is driven entirely by the OCM: it turns out that many of the failure types correspond closely to behaviours described informally in the ATC literature, which to some degree corroborates the effectiveness of the approach.

6.2 Failure of scanning

The first obvious "non-action cycle" through Figure 4 is where the operator fails to take the transition from scanning to monitoring p. (They may of course monitor and act on other pairs than p.) This class of behaviours is described formally by the formula $\neg \Diamond s_p$.

This behaviour type is considered to be a failure of scanning. Scanning methods that predispose the operator to not monitoring possible conflicts include tunnel vision, encystment and vagabonding [Reason, 1987]. Tunnel vision can be described by analogy with scanning with a searchlight: the operator looks only at a small section of the display at a time, without taking notice of what is happening elsewhere on the display. Encystment is where the operator focusses on a single problem and ignores everything else. Vagabonding is where the operator is skipping from problem to problem without spending enough time on each, and could be caused for example by high workload. Other possible causes of failure of scanning include poor UIs (e.g. low resolution displays, ambiguous display of flight route information).

6.3 Persistent mis-classification

The next "non-action cycle" through Figure 4 is where the operator monitors the pair p but persistently passes straight back to scanning after classifying the pair. There are two cases to consider:

- Persistent mis-classification: every time the operator monitors the pair they classify it as a non-conflict (and thus return immediately to scanning)

  $\Diamond s_p \wedge \Box(s_p \Rightarrow \bigcirc c_p^3)$

  $\Diamond s_p$ is included above to avoid overlap with failure of scanning.

- Persistent mis-prioritisation: the operator correctly classifies the conflict, but always defers determining corrective actions (see Section 6.4 below).

There are many cases where a controller might monitor a conflict pair and yet not classify it as a conflict: for example, because of a failure to "project forward" correctly, or a case of mistaken identity, or because the operator believes that have already dealt with the problem.

Under normal circumstances, it is likely that the operator will recover from single a mis-classification error, since the rule-based ProjectForward process will eventually win out over LookupMemory as the distance between the pair decreases. However, having once classified a pair as a nonconflict, the weight of the memory-based part of the classification process (LookupMemory) is likely to be strengthened, perhaps to the point where the operator's perception is distorted by the mistaken belief that the pair is not in conflict. (Experiments with the ATC simulator have demonstrated that prior exposure to repeated "near miss" problems at a particular waypoint increases the likelihood that operators will classify a similar episode as a non-conflict, even if it actually is a conflict.)

Other factors which decrease the likelihood of recovery from mis-classification include distractions that prevent return to monitoring the pair, prior exposure to a large number of non-conflicts that are very similar to the current event, and workload.

6.4 Persistent mis-prioritisation

The "persistent mis-prioritisation" failure described above can be formalised as follows:

$$\Box(c_p^1 \Rightarrow \bigcirc s) \wedge \Diamond c_p^1$$

To avoid overlap with the "persistent mis-classification" failure, we include the condition that at some time the operator correctly classifies the pair.

It is often the case that an operator will be working on one problem when another becomes apparent. If this problem is of a higher priority than the current one, it is often the best course of action to change the focus to the more urgent problem.[6] Mis-Prioritisation can arise for example because of a failure of the Prioritisation process, or because the time planned for taking corrective action is mis-calculated, mis-stored or mis-retrieved from memory.

6.5 Defer action for too long

This is the case where the operator has determined corrective actions but defers taking them for too long; formally:

$$\Diamond(d_p^1 \vee d_p^2) \wedge \Box((d_p^1 \vee d_p^2) \Rightarrow \bigcirc s)$$

The formalisation covers both the "effective plan" and "ineffective plan" cases.

This sort of error is often associated with failure of memory, where the operator has intended to return to a task, but for some reason the intention has not been retrieved from memory (more likely when there is a very high workload). It can also occur if the operator mis-calculates when to take corrective action.

This error can also occur if an action is intended but not successfully invoked (e.g. because the user slips when selecting from the speed menu, and clicks outside the menu without realizing it).

6.6 Unable to take action

The next case is where the operator has determined corrective actions but for some reason is unable to take them. For the ATC simulator, this case would only arise if the operator is unable to use the UI, say because of a hardware failure (such as the mouse cable breaking) or because they have trouble selecting the aircraft or using the speed menu. In a "real" ATC system even such UI failures would be recoverable, since radio communications would normally still be available to the controller (although of course if the display failed altogether the controller's task will be a lot harder!)

[6] Strictly, transitions of the form $c_p \rightarrow s_q$ should be added to the OCM to cover this "memory cued problem monitoring" case; if this is done, then $\bigcirc s$ will need to be replaced by $\neg \bigcirc(d_p^1 \vee d_p^2)$ in the failure mode.

This failure is harder to express in our formalism; the following trick says that the OCM trace did not progress past determining the action plan:

$$\Diamond((d_p^1 \vee d_p^2).\,\text{false})$$

Note that the case where the operator is unable to take action on p, but continues interaction with the system, is a subcase of "defer action for too long".

6.7 Discussion

The above discussion illustrates how the formalism can be used to describe task failures precisely and concisely, and how the Operator Choice Model can be used decompose the high-level task-failure conditions into subtypes of operator behaviour that can be analysed separately. A partial validation of this approach to human-error identification and classification is that we have subsequently found that many of the behaviour types predicted above correspond closely to those described by domain experts (e.g. use of inefficient scanning techniques as a key contributor to loss of situational awareness [Redding and Seamster, 1996]).

The formality of the OCM means that the above categories can be argued to be complete and non-overlapping. The approach is thus effective in breaking down the high-level failure (in this case, non-response) into subclasses of operator failure that can analysed separately. It is an open research problem to determine what additional constraints (if any) need to be placed on the OCM in order to formally prove that the lower-level behaviours in Section 6.2-6.5 cover all of the "non-response" behaviours defined in Section 6.1.

In the above treatment no distinction was drawn between the "effective plan" (d_p^1) and "ineffective plan" (d_p^2) cases. At this point the reader may be asking whether the OCM should try to distinguish the two cases at all. In fact, the distinction is probably not at all important for this level of analysis; it does however become important when associating likelihoods with different types of behaviour.

Returning to the question of limitations of the model, it will be observed that certain kinds of error cause are not well represented. Common-mode causes such as workload and distractions are not directly expressible over the model. A more specific example is "pair confusion", whereby the operator is thinking about one pair but actually looking at another. As with any modelling approach, certain compromises have to be made in order to make analysis tractable.

7 Conclusions and further work

In summary, this paper has introduced a new approach to formalising analysis of human errors in

human-computer interaction, and to decomposing operator failure modes into subtypes. The approach improves on existing human-error identification techniques by providing a richer modelling framework and being more expressive. This is particularly important for analysis of tasks involving highly interleaved concurrent, ongoing activities, since it enables errors to be modelled as behaviours rather than as events.

The formal models use a combination of CSP and temporal logic – two techniques familiar to computer scientists and applicable to a wide class of interactive systems. The paper illustrates a top-down analytic approach to human-error classification for a highly simplified Air Traffic Control task. Although the task studied is relatively simplistic from a real ATC operator's viewpoint, the example illustrates many of the issues that would be involved in analysis of a "real-life" task. Perhaps the most challenging aspect of applying the approach to real-life systems will be in developing appropriate models of their environments: the analysis for the example given here was relatively simple because the environment (the ATC simulator) was entirely deterministic.

The approach is part of a broader approach being developed for quantitative Human Reliability Assessment (HRA). The models will be extended with stochastic information such as probabilities and time-durations of transitions. Continuous Time Probabilistic Automata [Hung and Chaochen, 1999] provide the conceptual framework for the extensions. Simulator-based experiments are being used to calibrate the stochastic model. At this point the feasibility of the calculations involved – and credibility of quantitative results – is yet to be established, but even so, the approach is expected to help analysts express and reason about HCI design options.

At this stage, independent validation of the effectiveness of the approach is limited to feedback we have received from domain experts (that the behaviour types that our models predict do indeed occur) and from our cognitive psychology colleagues (that the models aid in experimental design).

Another area for future work concerns investigation of the extent to which the approach can usefully be pushed down to finer levels of granularity in the analysis of errors associated with cognitive processes. Propagation of errors through the cognitive model in Section 3 could clearly be analysed further, exploiting CSP's facility for expressing concurrent processes.

In another direction, we shall use model checkers [Clarke *et al.*, 2000] to try to prove that our decomposition is logically complete. More generally, such tools may be useful during error classification for identifying behaviours that have not yet been considered.

Acknowledgements

The authors gratefully acknowledge the collaboration of Andrew Neal and Mike Humphreys from the University of Queensland's Key Centre for Human Factors and Applied Cognitive Psychology in the development of the cognitive model in Section 3. Andrew Neal is also our resident ATC domain expert, and the Key Centre provided the ATC simulator. The Key Centre's Shayne Loft is designing and supervising the experiments that are being used to validate the cognitive model and to calibrate the error model. Finally, the ARC's support for the research reported here – by way of a Small Grant in 2000 and a Large Grant in 2001-3 – is gratefully acknowledged.

References

[Ben-Ari *et al.*, 1983] M. Ben-Ari, Z. Manna, and A. Pnueli. The temporal logic of branching time. *Acta Informatica*, 20:207–226, 1983.

[Clarke *et al.*, 2000] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 2000.

[Connelly *et al.*, 2001] S. Connelly, P. Lindsay, A. Neal, and M. Humphries. A formal model of cognitive processes for an air traffic control task. Software Verification Research Centre TR01-31, The University of Queensland, August 2001.

[Hoare, 1985] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[Hung and Chaochen, 1999] Dang Van Hung and Zhou Chaochen. Probabilistic duration calculus for continuous time. *Formal Aspects of Computing*, 11:21–44, 1999.

[Johnson, 1997] C. Johnson. Reasoning about human error and system failure for accident analysis. In S. Howard, J. Hammond, and G. Lindgaard, editors, *Human-Computer Interaction INTERACT '97*, pages 331–338. Chapman and Hall, 1997.

[Kallus *et al.*, 1999] K. Kallus, D. van Damme, and A. Dittmann. Integrated task and job analysis of air traffic controllers – phase 2: task analysis of en-route controllers. Technical Report HUM.ET1.ST01.1000-REP-04, European Organisation for the Safety of Air Navigation (Eurocontrol), October 1999.

[Kirwan, 1990] B. Kirwan. Human reliability assessment. In *Evaluation of Human Work*, chapter 28. Taylor and Francis, 1990.

[Kirwan, 1992] B. Kirwan. Human error identification in human reliability assessment. part 1: Overview of approaches. *Applied Ergonomics*, 25(5):299–318, 1992.

[Leadbetter *et al.*, 2000] D. Leadbetter, P. Lindsay, and A. Hussey. Formal modelling of

an air-traffic control simulator. Technical Report 00-25, Software Verification Research Centre, The University of Queensland, Brisbane 4072, Australia, December 2000. http://svrc.it.uq.edu.au/Bibliography/svrc-tr.html?00-25.

[Leadbetter *et al.*, 2001] D. Leadbetter, A. Hussey, P. Lindsay, A. Neal, and M. Humphreys. Towards model-based prediction of human error rates in interactive systems. In *Proc 2nd Australasian User Interface Conference (AUIC 2001)*, number 5 in Australian Computer Science Communications vol. 23. IEEE Press, 2001.

[Leveson *et al*, 1997] N. Leveson *et al.* Final report: Safety analysis of air traffic control upgrades. NASA technical report, 1997. http://sunnyday.mit.edu/papers/dfw2.pdf.

[Leveson, 1995] N. G. Leveson. *Safeware: System Safety and Computers.* Addison-Wesley, 1995.

[Oestereich, 1999] B. Oestereich. *Developing Software with UML.* Addison Wesley, 1999.

[Palanque *et al.*, 1997] P. Palanque, R. Bastide, and F. Paterno. Formal specification as a tool for objective assessment of safety-critical interactive systems. In S. Howard, J. Hammond, and G. Lindgaard, editors, *Human-Computer Interaction INTERACT '97*, pages 323–330. Chapman and Hall, 1997.

[Reason, 1987] J. Reason. Generic error-modelling system (GEMS): A cognitive framework for locating common human error forms. In J. Rasmussen, K. Duncan, and J. Leplat, editors, *New Technology and Human Error*, chapter 7, pages 63–83. John Wiley and Sons Ltd., 1987.

[Redding and Seamster, 1996] R.E. Redding and T.L. Seamster. Cognitive task analysis of air traffic control instruction to identify rule-based measures of student simulator performance. In *Proceedings of the Human Factors and Ergonomics Society 40th Annual meeting*, pages 269–273, 1996.

[Seamster *et al.*, 1997] T.L. Seamster, R.E. Redding, and G.L. Kaempf. *Applied Cognitive Task Analysis in Aviation.* Aldershot, 1997.

[Svenson, 1989] O. Svenson. On expert judgements in safety analysis in the process industries. *Reliability Engineering and System Safety*, 25:219–256, 1989.

[Swain and Guttmann, 1983] A. Swain and H. Guttmann. *Handbook of Human Reliability Analysis with Emphasis on Nuclear Power Plant Applications.* NUREG/CR-1278, U.S. Nuclear Regulatory Commission, 1983.