TECHNICAL REPORT

No. 00-33

Towards Model Based Prediction of Human Error Rates in Interactive Systems

David Leadbetter, Peter Lindsay, Andrew Hussey, Andrew Neal[1] and Mike Humphreys[1]

[1]Key Centre for Human Factors and
Applied Cognitive Psychology
The University of Queensland

November 2000

Towards Model Based Prediction of Human Error Rates in Interactive Systems

David Leadbetter, Peter Lindsay,
Andrew Hussey, Andrew Neal[1] and
Mike Humphreys[1]

[1]Key Centre for Human Factors and
Applied Cognitive Psychology
The University of Queensland

Abstract

Growing use of computers in safety-critical systems increases the need for Human Computer Interfaces (HCIs) to be both smarter — to detect human errors — and better designed — to reduce likelihood of errors. We are developing methods for determining the likelihood of operator errors which combine current theory on the psychological causes of human errors with formal methods for modelling human-computer interaction. We present the models of the HCI and operator in an air-traffic control (ATC) system simulation, and discuss the role of these in the prediction of human error rates.

Keywords: HCI, ATC, cognitive model, error rate.

1 Introduction

A human-computer interface is safety-critical when the potential arises for injury or loss of life from defects in the design of the HCI. Safety has frequently been compromised and lives have been lost because of operator errors caused by HCI design deficiencies (e.g., see [12]).

Existing models of human error do not provide a precise specification of the conditions leading to error, or the mechanisms responsible for error. The complexity of modelling human behaviour in complex real world systems compounds this problem. We are developing a formal model of cognition in a case study involving a simplified ATC simulation based on psychological theories of human error. The formal operator model is integrated with a model of the ATC HCI to enable identification of sources of operator error and corresponding HCI features that diminish error.

The ATC Case Study

The case study involves a simulated, simplified air-traffic control task. The simulated air sector is presented to the air traffic controller via a graphical HCI.

The simplified task deals with air-traffic control in an en-route sector. Simplifications present in the task include:

- aircraft move only in two dimensions (no altitudes)

- aircraft travel on ‘rails’
- instant speed changes
- instant course changes
- operations for changing aircraft speed only (no re-routing, etc)
- obedient pilots (i.e. respond to operator instructions)

A fabricated screen shot of the simulation HCI is provided in Figure 1. Within the figure aircraft are represented as dots moving along defined flight paths.

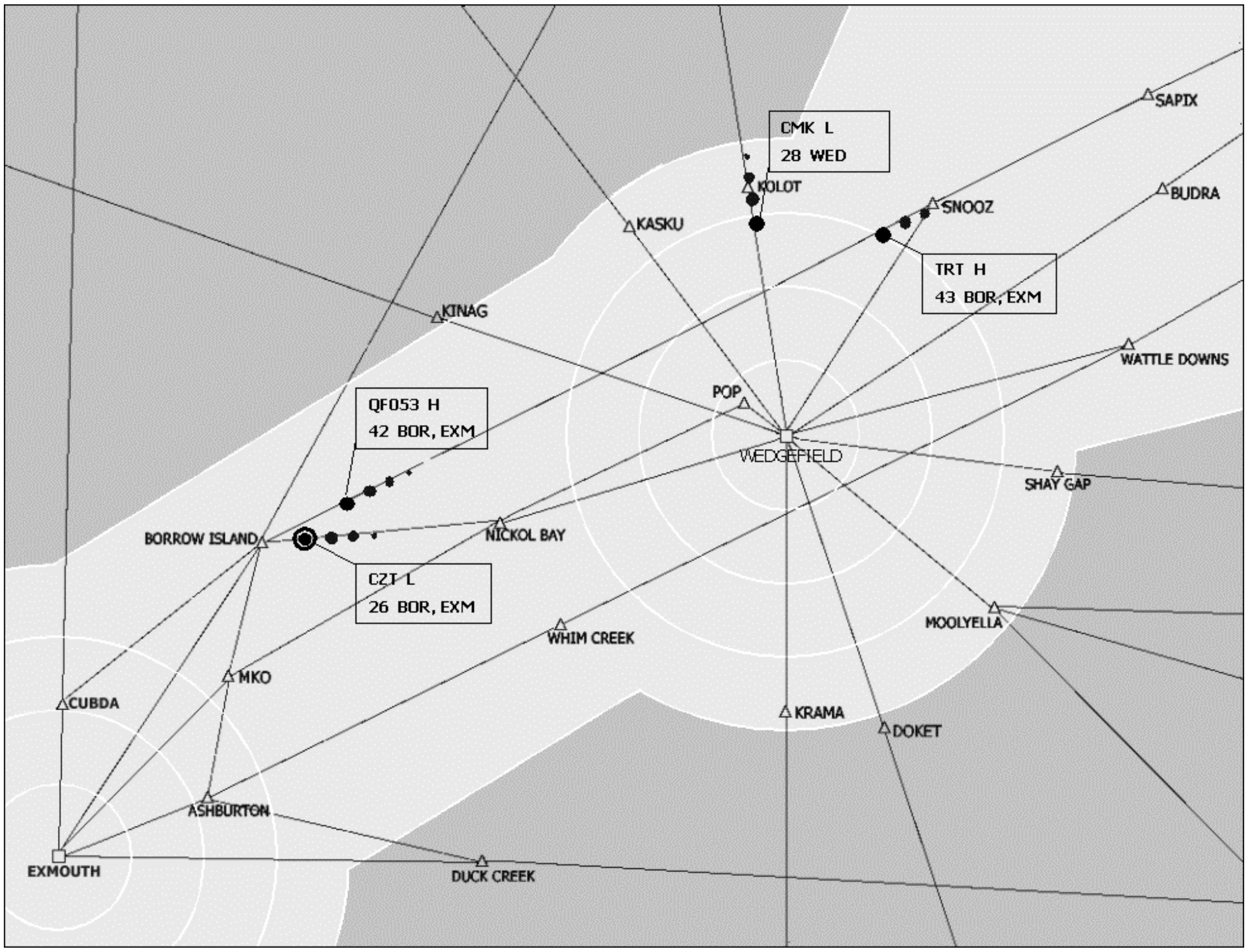

Figure 1: A sector of controlled airspace

The case study has been modelled in three separate sections:

- The ATC core system,
- The ATC Human-Computer Interface, and
- The Operator model.

The coupling between the parts of the ATC system is illustrated in Figure 2. The models of the ATC HCI and operator presented in this paper represent the results of the first cycle in an iterative modelling process. As such, the models include many simplifications, primarily through the use of abstraction so that details of the system parts can be ignored at this stage. Such details will be added during later iterations of the modelling process. The ATC core system is not described in this report (see [11]).

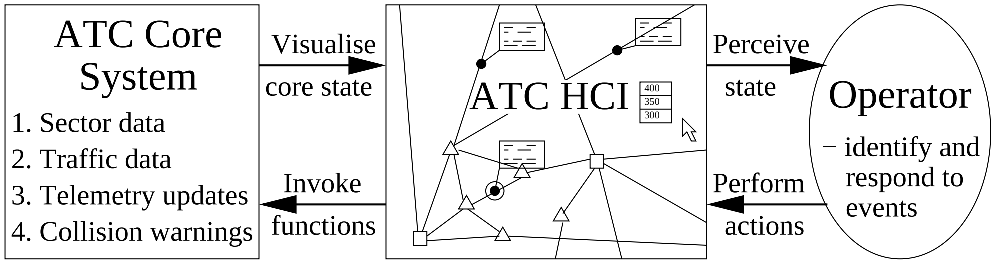

Figure 2: Relationship between system, HCI, and operator models

The ATC Human-Computer Interface

The HCI visualises the state of the underlying simulation and provides a basic range of operations for acting on it - for simplicity, operations are only provided to allow the aircraft speeds to be changed; more advanced operations, such as changing the flight plan are not provided. Figure 1 illustrates what the visualisation provided by the HCI may look like: aircraft are represented as dots over a graphical representation of the layout of the waypoints and flight routes in the sector; basic aircraft details (callsign, speed, etc) are displayed in boxes attached to the aircraft dots.

The ATC Operator

The primary task of the operator is to ensure that the aircraft moving through the sector remain separated by a defined “minimum separation” distance. The operator must use the operations provided by the HCI to avoid violations of minimum separation that appear likely to occur. Real ATC controllers are also concerned with other objectives such as ensuring that aircraft move efficiently through the sector with minimal delay and disruption. Such concerns are not considered here.

2 Background and overview

Although considerable progress has been made in understanding the task conditions that are likely to lead to human error, these traditional approaches do not allow precise formulations of error to be developed. Our approach is based on a number of recent advances in cognitive psychology, including the development of connectionist models of memory [14], and mathematical specifications describing the input-output functions required to model different tasks [7]. These advances allow us to develop precise specifications of the conditions and processes that lead to a wide range of human memory errors.

A substantial body of related work deals with the integrated modelling of the operator with the computer aspects of a system. As in Duke et al. [4], we advocate the integration of a cognitive model of the operator. The cognitive models presented in this paper differ from the work of Duke et al. because they describe the controller's cognitive functions (rather than constructions of explicit cognitive subsystems) and are based on new psychological theories. Our work is more abstract and, at this stage, arguably less formal than that of Duke et al., but thereby more suited to use by psychological researchers. Furthermore, we believe the combination of modelling notations used in this research to be unique. The notations used here were chosen both because they enable an appropriate and useful level of abstraction to be achieved for each of the models and because they are easily understood. Butterworth et al. [2] describe Programmable User Models (PUMs) using a state-based notation. PUMs focus on the user's model of the system, with formalisation of the information needed and actually acquired by a user and the HCI operations available to the user. PUMs are similar to the HCI model constructed in this paper but do not incorporate elements of the physical user actions and system presentation. Palanque et al. [17] have used Petri-nets to model tasks, systems and the user's view of the system. Their approach uses an object-oriented extension of the Petri-net notation and enables analysis of incompatibilities between the models. However there is no modelling of either the HCI presentation or the cognitive processes of the user. The method has been applied to analysis of an air-traffic control system [18].

In Section 3, we provide a model of the HCI for the ATC system simulation which combines the UAN [6] and Z notations [20]. Section 4 gives a model of the operator's cognitive process using Statecharts [5].

3 The ATC HCI model

The ATC HCI is modelled using an integrated approach, blending a formal model of the interface state with a notation for describing the user actions.

3.1 The Underlying Interface State

The underlying interface state defines what information is contained in the HCI (but not how it is presented). The most significant information in the HCI is the air traffic in the sector. Each aircraft is represented in the HCI as a view (abstractly modelled using the given type *AircraftView*). Other operational information is included in the HCI state. The current selection is included within the HCI state (modelled as a set allowing either no aircraft or a single aircraft to be selected). The *speedMenu* used by the operator to instruct the selected aircraft to change speed is also included (modelled using the given type *SpeedMenu* - we assume that the menu corresponding to a particular view is generated in the following way: $makeMenu(aircraft(view))$), and the set of aircraft that are currently in a separation violation (*warnings*).

ATCInterface

$views : \mathbb{P}\, AircraftView$
$selected : \mathbb{P}\, AircraftView$
$speedMenu : SpeedMenu$
$warnings : \mathbb{P}\, AircraftView$

$selected \subseteq views$
$\# selected \leq 1$
$warnings \subseteq views$

3.2 The User Action Notation

The remainder of the HCI model is defined using the User Action Notation (UAN) combined with fragments of Z. UAN is a simple notation for describing “the behaviour of the user and the interface as they perform a task together” [6]. UAN provides symbols for user actions that describe the interaction between the user and the computer while performing tasks (such as moving and clicking the mouse), and feedback symbols that describe feedback from the interface (such as what is displayed on the screen).

Some additional feedback symbols for the various forms of highlighting the aircraft views in the ATC HCI are defined in Figure 3. The two forms of highlight provided by these symbols are not exclusive; it makes sense to apply both at the same time.

Symbol	Meaning
$!_{\mathsf{S}}$	selection highlight (a circle is drawn around the aircraft dot in the aircraft view)
$!_{\mathsf{W}}$	warning highlight (a different colour is used for the aircraft view)
$-!_{\mathsf{S}}$	no selection highlight
$-!_{\mathsf{W}}$	no warning highlight

Figure 3: Additional feedback symbols for highlighting aircraft views

3.3 Visualisation of the ATC System State

Visualisation of the ATC system state consists of displaying the aircraft views at the appropriate positions on the screen. We assume that the position of the aircraft represented by an aircraft view may be obtained in the following way: $position(aircraft(view))$. Furthermore, we assume that this real world position is converted to the corresponding screen coordinate using the function $convert()$.

The HCI state (as defined in *ATCInterface*) is visualised in the following way (using the feedback symbols of UAN).

The aircraft views are displayed in the appropriate positions:

$$\begin{array}{l} \forall\, acView : views \bullet \\ \quad @convert(position(aircraft(acView))) \\ \qquad \mathsf{display}(acView) \end{array}$$

The selection highlight is applied to any selected view:

$$\begin{array}{l} \forall\, acView : selected \bullet acView!_{\mathsf{S}} \\ \forall\, acView : views \setminus selected \bullet acView-!_{\mathsf{S}} \end{array}$$

The warning highlight is applied to every aircraft view included in the warnings:

$$\begin{array}{l} \forall\, acView : warnings \bullet acView!_{\mathsf{W}} \\ \forall\, acView : views \setminus warnings \bullet acView-!_{\mathsf{W}} \end{array}$$

3.4 The User Actions

The interactions between the user and the HCI when performing various actions are defined below. In these definitions, the coupling with the underlying interface state is implicit in the use of common attribute names. Priming of attribute names is used as in Z to distinguish the before and after values of an attribute. For example, the first user task defined below, *selectAircraft*, involves the interface state attribute *selected* which is modified by the action (indicated by assigning to *selected'*).

Task: *selectAircraft*

The operator selects an aircraft by moving the mouse over the appropriate aircraft view and clicking the left mouse button; the highlighted aircraft view changes accordingly:

User Action	Interface Feedback	Interface State
$\sim$[aircraft_view]		
$\mathrm{M}_L \vee \wedge$	$\forall\, acView : selected \bullet$	$selected' = \{$ aircraft_view $\}$
	$acView\,{-}!_{\mathsf{S}}$	
	aircraft_view $!_{\mathsf{S}}$	

Task: *changeAircraftSpeed*

The operator instructs the selected aircraft to change speed by opening the speed menu, navigating the menu to the desired speed, then selecting the desired speed:

openSpeedMenu $\fatsemi$ *navigateSpeedMenu* $\fatsemi$ *selectSpeed*

Note that the '$\fatsemi$' symbol used above is the task interrupt symbol. If the user interrupts *changeAircraftSpeed* the effect is: erase(*speedMenu*)

Subtask: *openSpeedMenu*

If an aircraft view is selected, the operator can open the speed menu by clicking the right mouse button:

User Action	Interface Feedback	Interface State
$selected \neq \varnothing$:		$\exists_1\, acView : selected \bullet$
($\sim$[x,y] $\mathrm{M}_R \vee \wedge$)	@ x,y display($speedMenu'$)	$speedMenu' =$
		$makeMenu(aircraft(acView))$

Subtask: *navigateSpeedMenu*

The operator navigates within the speed menu by moving the mouse in and out of the lines in the menu; the highlighted menu line changes appropriately as the operator navigates the menu:

User Action	Interface Feedback
$\sim$[line **m** in *speedMenu*]	line **m** !
($\fatsemi$ [line **m** in *speedMenu*]$\sim$ $\fatsemi$	line **m** $-$!
$\sim$[line **n** in *speedMenu*])*	line **n** !

If the user interrupts *navigateSpeedMenu* the effect is: erase(*speedMenu*)

Subtask: *selectSpeed*

The operator selects a speed from the speed menu when the mouse is over the appropriate speed line by clicking the left mouse button:

User Action	Interface Feedback
$\sim$[line **m** in *speedMenu*] :	
$\mathrm{M}_L\vee\wedge$	erase(*speedMenu*)

4 The ATC Operator Model

The operator model details the high level cognitive processes of the ATC operator, and avoids details of the low level mechanics of human memory.

The model is a memory based model, featuring a number of different memories. These memories, and their contents that are of interest are briefly described.

4.1 Operator Memories

The cognitive model of the simulation operator involves a number of different memories. The most important of these memories are the operator's episodic and short-term memories. These memories are intended to capture the functionality of human memory (its dependence on cues, number of rehearsals, recency of occurrence and capacity) in a parsimonious manner and are not intended as psychological or physiological hypotheses about the fundamental structure of human memory.

Episodic memory

The operators episodic memory is used for recording the simulation episodes experienced by the operator. It contains a sequence of event relations (defined in more detail below) each describing a previous simulation experience. The memory for a particular event relation is cued by the information presented on the screen such as the aircraft, call sign, location, position in relation to other aircraft, etc. This memory has a long term or semantic component (it is affected by the memory for similar episodes which occurred on previous days), an intermediate term component (it is affected by the number of times the current event relation has been retrieved and stored during the scanning process), and a short-term component (it is affected by the recency of the last storage). The episodic memory has a large capacity.

Short-term memory

The operators short-term memory is used to temporarily record information of recent relevancy. This includes event relations for events recently experienced and their associated priorities. There is no specific cue for these event relations so recall is determined by recency. The short-term memory has a very limited capacity.

4.2 The event relation

The event relation is the main form of information stored in each of the above two memories in the cognitive model. An event relation takes the following form:

$$event(aircraft\ attributes, context, classification, time, action)$$

The various elements of an event relation are as follows:

event	The type of event: nonevent, converge, or overtake.
aircraft attributes	The aircraft attributes: e.g. callsign, type, speed.
context	The context in which the event occurs: e.g. time of day, event position.
classification	The event classification: conflict, nonconflict, or ?
time	The latest time at which it is projected that corrective action can be taken
action	The corrective action to be taken.

Consider, for example, the following event relation:

converge$((\{QF053, H, 42\}, \{CZT, L, 26\}),$
"Approaching Borrow Island en-route to Exmouth airport",
conflict, $11{:}23{+}10{\pm}1, ?)$

This describes a convergence event involving the two aircraft - QF053 (a heavy aircraft travelling at 420knots) and CZT (a light aircraft travelling at 260knots) - as they approach the waypoint 'Borrow Island' en-route to 'Exmouth airport'. The event has been classified as a conflict, so a separation violation is expected to result as these aircraft converge on Borrow Island. In order to avoid this violation corrective action needs to be taken sometime between 11:32am and 11:34am (10 minutes after 11:23am give or take 1 minute). The event relation does not provide a corrective action.

4.3 The Cognitive model

The operator's cognitive process consists of a multi-stage cycle involving scanning for events, projecting the event forward in time, prioritising the event, making a decision, and performing the decided action. These stages (modelled using statecharts in Figure 4) are briefly described here - for a more detailed description see [11].

Scanning

Scanning for events involves monitoring the HCI until the operator matches the arrangement of some aircraft on the screen with one of the known event geometries. An event relation identifying the event is obtained - the operator retrieves a matching relation from memory if possible, and cognitively produces a new relation: the most compelling of these two relations being chosen to identify the event. If the controller remembers acting on the event previously they restart the scanning process. Otherwise the event relation is stored in memory and the process continues.

Project Forward

The operator projects the event forward in time to estimate the time at which action must be taken to resolve the event, and to classify the event to determine whether operator intervention is needed. An event must be projected if both of these are unknown, and is skipped when immediate action is required (when the time of action is now or passed). After the projection the process continues to prioritisation.

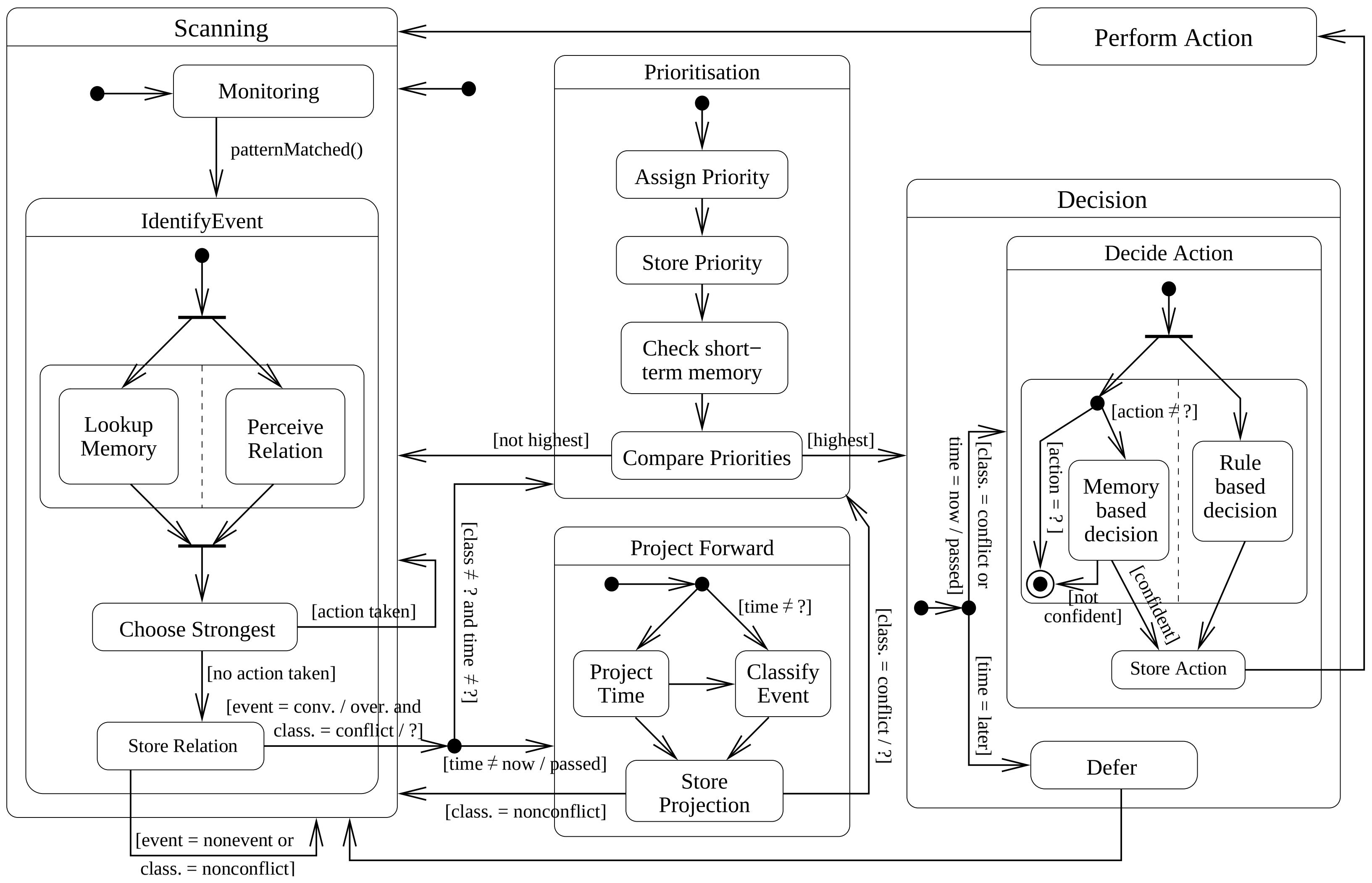

Figure 4: The control flow model of the ATC simulation operator

Prioritisation

The operator uses event priorities to attend to events in an appropriate order. An event is assigned a priority which is compared with the priorities of other events in short-term memory. If the event has the highest priority the operator continues to the decision process, otherwise they return to scanning (for a higher priority event).

Decision

If the event time is later the operator may defer deciding an action until a later time, in which case they return to scanning. Otherwise a corrective action is decided immediately. The operator uses two approaches to decide on the corrective action. These are: memory-based decision (if the event relation was retrieved from memory and included an action); and rule-based decision. The memory-based action is used if the operator has confidence in it, otherwise the rule-based action is used. The operator stores the decided action (with the event relation) and proceeds to perform the action.

Perform Action

The operator performs the action through interactions with the ATC HCI, where those interactions are as defined in Section 3.4. Once the action has been performed, the operator returns to scanning.

5 Human Error Rate Analysis

5.1 Method

Human errors may manifest themselves as errors of perception, generation of actions, or action performance. Errors of perception and action performance relate to cognitive tasks concerning direct interaction with the HCI. In our model of the operator's cognition, perception relates to the Monitoring task, and action performance relates to the Perform Action task. Generation of actions involves all of the remaining cognitive tasks.

There is potential for errors to occur in every task (and subtask) in the cognitive model. These errors are manifest primarily as an incorrect output from the task (with respect to the task input), but may also be manifest in other ways. For example, for the tasks that store the event relation into memory, the error could be related to memory (such as storing the relation inaccurately) rather than the task output.

By using the cognitive model as the basis for the analysis, the granularity of the analysis is diminished, an advance over existing techniques, and the ability to focus design interventions on particular cognitive tasks is facilitated.

Each different type of error that occurs in a cognitive task needs to be considered individually. For every different error mode of a cognitive task there is an associated base error rate. This base error rate is a probability that defines the likelihood of the input/output function producing one of the outputs associated with that error mode for the input to the task if all external factors are ignored.

External factors have a multiplicative effect on the base error rates associated with the different error modes of a task; this is similar to the approach used in several existing human error quantification methods, such as THERP (Technique for Human Error Rate Prediction) [10]. These multipliers are also specific to each task, such that, for example, workload has a different effect on the error rates of different tasks. The multipliers may either increase or decrease the base error rates of a task, implying that different external factors can either have a detrimental or beneficial effect on the task respectively.

The derivation of these error rates raises questions about the quantitative accuracy of the rates that can only be validated by comparison with (and calibration from) experimental testing. We are currently in the process of performing such experiments.

Redmill and Rajan [19] has produced a categorised list of external factors:

Task demands and characteristics: Frequency, workload, duration, interaction with other tasks, perceptual, physical, memory, attention required, vigilance required.

Instructions and procedures: Accuracy, sufficiency, clarity, level of detail, meaning, readability, ease of use, applicability, format, selection and location, revision.

Environment: Temperature, humidity, noise, vibration, lighting, work space, movement restriction, operator control of environment.

Stresses: Time pressure, workload, fatigue, monotony, isolation, distractions, shift work incentives.

Individual: Capacities, training and experience, skills and knowledge, personality, physical condition, attitudes, motivation, risk perception.

Socio-technical: Staffing adequacy, work hours and breaks, resource availability, social pressures, conflicts, team structure, communications, roles and responsibilities, rewards and benefits, attitude to safety.

Displays and controls: Compatibility, ease of operation, reliability, feedback, sufficiency, location, readability, identification, distinctiveness.

Potentially hazardous errors can be identified by using a combination of HAZOP [15] and Functional Failure Analysis (FFA) for the cognitive model (e.g., [12]). The FFA examines the functional operation of tasks in the operators cognitive process to identify errors arising from information processing faults. The HAZOP considers errors arising from perturbations in the information flow: lists of standard keywords (such as “no”, “less”, “more”, “other”) are applied to information flows in the operator’s cognitive process. The HAZOP enables faults to be traced back to their cognitive source (as indicated by the FFA), and forward to an accident. Other human related errors involve the HCI displays and controls. An analysis of these errors is informed by the HCI model.

Design interventions for safety-critical systems provide guidance for how to design the user-interface such that particular operator errors may be diminished. The effect of each design intervention is task specific (some design interventions may have no effect on some cognitive tasks). Design interventions for safety-critical systems make use of usability guidelines such as those found in [1, 3, 16], taking account of the special requirements for safety-critical systems. Design interventions may also draw on more structured repositories of design knowledge such as the “pattern” languages recorded by [9, 13]. Patterns provide both the context in which a design solution applies, and examples of the successful application of the solution in industry. Design interventions for cognitive tasks could be similarly structured as patterns, with clearly defined cognitive contexts and examples in which the solutions have been applied.

5.2 ATC case study

Error Sources

The cognitive tasks in the ATC simulation are one of the main sources of error in the cognitive model (the other main source of error in the cognitive model being the control flow of the inter-task transitions). We consider only the following cognitive tasks in this example: lookup memory, classify event, rule-based decision.

A HAZOP and FFA analysis of the cognitive model of the ATC simulation operator is given in detail in [8]. For example, the Lookup Memory task retrieves a matching event relation from episodic memory and outputs with that event relation the strength of the match. The FFA concerning the output strength of the Lookup Memory task is as follows (the table shows only the local effects of the each failure mode):

Failure Mode	**Effects**
under confidence	Output strength is low
over confidence	Output strength is high
unknown confidence	No strength is output
false confidence	A strength is output when there is no match

Effect Matrix

External factors effect the error rates for different error modes of the cognitive tasks in various ways – this includes the possibility that an external factor has no effect on a particular error mode. We record the effect of external factors on cognitive tasks using an effect matrix.

The sample matrix illustrated here indicates only the general effect of each external factor on each cognitive task – that is how an increase in the external factor effects the error rate, and how a decrease in the external factor effects the error rate, or that the external factor has no effect on the error rate. Ultimately, each cell in the matrix would be replaced with a concrete measure of the effect, possibly as a function (from the level of the factor to a multiplier), or as a table (of factor levels and the associated multiplier).

We show a simplistic example effect matrix for the cognitive tasks discussed in the sections above.

	Cognitive Task		
External Factor	Lookup memory	Classify event	Rule-based decision
Workload ('pressure')	↑w = ↑error ↓w = ↓error	↑w = ↑error ↓w = ↓error	↑w = ↑error ↓w = ↓error
Fatigue	↑f = ↑error ↓f = ↓error	↑f = ↑error ↓f = ↓error	↑f = ↑error ↓f = ↓error
Frequency (of event)	↑f = ↓error ↓f = ↑error	↑f = ↓error ↓f = ↑error	no effect

The central cell in this matrix describes the effect of operator fatigue on the error rate for the classify event task: an increase in fatigue sees a corresponding increase in errors when classifying events, while a decrease in fatigue sees a decrease in errors when classifying events.

Design interventions

Design interventions can be targeted at each individual cognitive task to reduce the base error rate(s) of the error sources associated with that task. Design interventions for HCI related cognitive tasks typically involve changes to the HCI design, or alternative HCI designs. The HCI model describes the design interventions actually used in the ATC HCI.

We give some example design interventions for the cognitive tasks listed above – the aim of these interventions being to lower the base error rates associated with these cognitive tasks.

Lookup memory: provide more applicable information via the HCI: for example, aircraft details (that were previously omitted), event details (that can be automatically generated).

Classify event: provide automated tools: for example, time based flight projection tools; or automatic identification of events (reducing/eliminating the need for the controller to manually classify events).

Rule-based decision: provide better training (at problem solving); decision support tools; online help and/or manuals.

These design interventions aim to reduce the error rate of the cognitive tasks in a number of ways. Illustrated here we see interventions concerned with increasing and improving the input information to the cognitive process (and hence to the individual cognitive tasks), and reduction in the amount of cognitive processing through provision of various tools. These changes in information flow between the HCI and the operator, and changes in cognitive processing imply changes in the cognitive model of the operator associated with the operator using the modified interface, changes that reduce the number or likelihood of failure modes, and may remove sources of failure.

Design interventions can also be targeted at each individual external factor to reduce the effect of that factor on the cognitive tasks (by reducing the multiplier values).

We give some example interventions for the external factors listed above - the aim of these interventions is to change the level of the external factor such that the multiplier is lowered. Unlike design interventions targeted at specific cognitive tasks, the design interventions that target specific external factors are commonly not HCI related. Rather, they manipulate the working conditions and environment of the operator related to each factor.

Workload: more controllers, less air-traffic, automated ATC decision systems (monitored by the controller), automatic identification of events (alerting the controller when action is required). *Workload is decreased.*

Fatigue: shorter shifts, more breaks, better training, ergonomic work environment setup, automated ATC decision systems (monitored by the controller), automatic identification of events (alerting the controller when action is required). *Fatigue is decreased.*

Frequency: reduce flight routes, limit carriers/etc, consistent (possibly abstract) representation of events. *Frequency is increased.*

These lists should be cross-checked to identify any interventions appearing in both lists (and any interventions appearing multiple times in either list). Clearly these interventions have a compound effect on the error rates, and may indicate those interventions that are most worth pursuing. For example, in the above lists, automated identification of events is a design intervention that applies to a number of external factors (workload and fatigue), and to the cognitive task classify event.

6 Conclusions

We have shown how formal models can be constructed of an air-traffic control simulation system HCI and operator. Z, as a proven notation for modelling state-based system, has been used to model the underlying state of the HCI. UAN, as a simple notation for describing the user and interface behaviour as a task is performed, has been used to abstractly model the user actions provided by the ATC system. Statecharts as a simple, diagrammatic notation, have been used to semi-formally model the operators cognitive process, such that the models can be easily validated by the psychologists.

A unique feature of this work is that it incorporates a model of the operator's cognitive state and process. This model identifies the information focussed on, and the basic psychological process employed by the operator.

Lastly, we briefly discussed the role of these models in a method for analysing operator error rates to enable more accurate prediction of risk due to operator error.

References

[1] Apple Computer. *Human Interface Guidelines: The Apple Desktop Interface.* Addison-Wesley, Reading, MA, 1987.

[2] R. Butterworth, A. Blandford, and D. Duke. The Role of Formal Proof in Modelling Interactive Behaviour. In P. Markopoulos and P. Johnson, editors, *Design, Specification and Verification of Interactive Systems, DSV-IS '98*, pages 113–128. Springer-Verlag, 1998.

[3] Microsoft Corporation. *The Windows Interface Guidelines for Software Design.* Microsoft Press, Redmond, WA, 1995.

[4] D.J. Duke, P.J. Barnard, D.A. Duce, and J. May. Syndetic modelling. *Human-Computer Interaction*, 13(4):337–393, 1998.

[5] D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming 8*, pages 231–274, 1987.

[6] H. R. Hartson. Temporal Aspects of Tasks in the User Action Notation. *Human-Computer Interaction*, 7:1–45, 1992.

[7] M. S. Humphreys, J. Wiles, and S. Dennis. Toward a theory of human memory: Data structures and access processes. *Behavioural and Brain Sciences*, 17(4):655–692, 1994.

[8] A. Hussey, D. Leadbetter, P. Lindsay, A. Neal, and M. Humphreys. A Method for Analysing Hazards and Error Rates Related to Operator Activities. Technical Report TR00-25, Software Verification Research Centre, The University of Queensland, July 2000.

[9] A. Hussey and M. Mahemoff. Safety Critical Usability: Pattern-based Reuse of Successful Design Concepts. In M. McNicol, editor, *4th Australian Workshop on Safety Critical Systems and Software* , pages 19–34. ACS, 1999.

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

[11] D. Leadbetter, P. Lindsay, A. Neal, and M. Humphreys. Integrating the Operator into Formal Models in the Air-Traffic Control Domain. Technical Report TR00-34, Software Verification Research Centre, The University of Queensland, 2000.

[12] N. G. Leveson. *Safeware, system safety and computers.* Addison-Wesley, 1995.

[13] M. J. Mahemoff and L. J. Johnston. Principles for a Usability-Oriented Pattern Language. In P. Calder and B. Thomas, editors, *OZCHI'98*, pages 132–139. IEEE Computer Society, 1998.

[14] J. L. McClelland and D. E. Rumelhart. *Parallel Distributed Processing.* MIT Press, 1986.

[15] Ministry of Defence. Draft Interim Defence Standard 00-58/1: A Guideline for HAZOP Studies on Systems which include a Programmable Electronic System. Directorate of Standardization, 1995.

[16] Open Software Foundation. *OSF/Motif Style Guide: Revision 1.2.* Prentice Hall International, Englewood Cliffs, NJ, 1993.

[17] P. Palanque and R. Bastide. Synergistic modelling of tasks, users and systems, using formal specification techniques. *Interacting with Computers*, 9(2), October 1997.

[18] P. Palanque, F. Paternò, and R. Bastide. Formal specifications for designing user interfaces of air traffic control applications. In S. Gnesi and D. Latella, editors, *Second International ERCIM Workshop on Formal Methods for Industrial Critical Systems.* IEE, 1997.

[19] F. Redmill and J. Rajan. *Human Factors in Safety-Critical Systems.* Butterworth Heinemann, 1997.

[20] J. M. Spivey. *The Z notation: a Reference Manual.* Prentice-Hall, 2nd edition, 1992.