SOFTWARE VERIFICATION RESEARCH CENTRE

DEPARTMENT OF COMPUTER SCIENCE

THE UNIVERSITY OF QUEENSLAND

Queensland 4072
Australia

TECHNICAL REPORT

No. 00-14

Modelling and Hazard Identification in an Air-Traffic Control User-Interface

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

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

April, 2000

Phone: +61 7 3365 1003
Fax: +61 7 3365 1533

Note: Most SVRC technical reports are available via anonymous ftp, from `ftp.cs.uq.edu.au` in the directory `/pub/SVRC/techreports`.

Modelling and Hazard Identification in an Air-Traffic Control User-Interface

Andrew Hussey, David Leadbetter, Peter Lindsay,
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 conducting a research project that aims to combine improved understanding of the psychological causes of human errors, with new formal methods for modelling human-computer interaction. In this paper, we formally model an air-traffic control system HCI and show how hazardous scenarios where controller error may lead to loss of separation between aircraft can be formally defined. Later in our project, the hazard scenarios will be used to develop metrics for determining the likelihood of operator errors for a given system state, without psychological expertise.

1 Introduction

1.1 Scope

This technical report presents the preliminary results of the SafeHCI project which concern the modelling of hazards in an air-traffic control (ATC) system. The SafeHCI project is an ARC Small Grant in collaboration with the Key Centre for Human Factors and Applied Cognitive Psychology. The next section outlines the broad aims of the SafeHCI project and provides background details.

1.2 Background

Computers are increasingly being used in safety-critical systems. Examples include interactive control systems for transport (road, rail and air), medical equipment, power stations and process plants. As society increasingly relies on computerised systems for safety, this is an area that will continue to grow in importance. One of the key factors influencing the safety of computerised systems is the design of the Human-Computer Interface (HCI). 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 (eg., see [11]). Current system safety standards - such

as in defence [2], railways [5], and process industries [9] - mandate or highly recommend formal (mathematical) modelling for safety aspects of hardware and software functionality because such models promote methodical, reproducible and auditable software development. However, the techniques used for evaluating and enhancing the safety of HCI designs are informal at best. This is largely attributable to:

- difficulty of formally modelling the interaction between operators and the system;
- lack of understanding of the psychological processes responsible for operator error;
- inability to formally specify the antecedent conditions that trigger those processes, and to estimate the resulting likelihood of errors; and
- lack of precise methods for mapping operator errors to design solutions.

We are conducting a project that will develop formal methods for evaluating the safety of interactive systems. These methods will incorporate mathematical models of operators, the HCI, and safety aspects of the system and the environment in which it operates. The project is, therefore, interdisciplinary in nature, involving collaboration between computer scientists (working at the Software Verification Research Centre: SVRC) and psychologists (working at the ARC Key Centre for Human Factors and Applied Cognitive Psychology). The specific aims of the project are to:

1. Develop formal models of the conditions giving rise to operator error and of human-computer interaction within a simplified ATC system.
2. Develop a risk analysis model for the simplified ATC system, to enable tracing of the effects of operator errors through to system mishaps, based on the models of operator error and the HCI.
3. Empirically test the formal models and calibrate the risk analysis model by quantifying the likelihood of operator error and system mishaps under different ATC operational profiles.
4. Evaluate the effectiveness of HCI design solutions generated by the methodology by modifying the HCI of the ATC system to reduce frequency of operator errors and system mishaps in accordance with the results of the methodology, and experimentally testing the result.
5. Validate the methodology by applying it to selected aspects of ATC systems currently used in the field.

This paper discusses progress made towards satisfying the first of these aims. To date, little progress has been made in this field, because existing models of human error have not provided a precise specification of the conditions leading to error, or the mechanisms responsible for error. This problem has been compounded by the complexity of modelling human behaviour in complex real world systems. One of the most commonly used methods for safety analysis is Failure-Modes and Effects Analysis (FMEA). Two examples in the HCI domain are the SHERPA (Systematic Human Error Reduction and Prediction Approach) and THERP (Technique for Human Error Rate Prediction) methods [10]. In such an FMEA, the designer inspects

components of the system and identifies possible failure modes and their potential effects using a "checklist" of common failure-modes (i.e., for the human component) [7]. Unfortunately, this approach is limited, because FMEA methods are subjective and informal, and the link between failure-modes and underlying psychological error mechanisms is weak. Instead, we are developing formal models of safety-critical systems that are based on psychological theories of human error.

Our approach will improve on existing interactive system safety analysis approaches, because the method will:

- operate on formal models of interactive systems so as to be precise, objective (auditable) and repeatable;
- be informed by psychological theory and thus enable deeper aspects of HCI design to be modelled and analysed;
- be empirically validated by psychological experiments and by application to industrial problems.

The advantage of formal models is that they enable human-computer interactions to be precisely specified and modelled at arbitrary levels of abstraction. Typically such models capture functions, tasks, information structures and environmental context using set theory (e.g., the Z notation [17]) and process notations (e.g., CSP [6]). Hussey has demonstrated the feasibility of using formal notations such as Object-Z [3] for producing system models [8]. Such models provide task descriptions in terms of sequences of actions and corresponding executions and can be analysed for memory-based errors such as selection of a wrong action sequence and autonomous execution errors [15]. Preliminary work has been conducted on applying these techniques to analyse safety aspects of HCI designs [7].

The formal models developed in the current project will incorporate mathematical models of human error. Although considerable progress has been made in understanding the task conditions that are likely to lead to human error [16], these traditional approaches do not allow us to develop mathematical formulations of error. Our approach is based on a number of recent advances in cognitive psychology, many of which have been pioneered by the project's partner investigator (Prof. Humphreys). Theoretical advances that we draw upon include the development of connectionist models of memory [14], and mathematical specifications describing the input-output functions required to model different tasks [13]. Empirical advances that we draw upon include new experimental techniques for studying prospective memory (e.g., [4]), and demonstrations of a wide variety of new types of memory errors due to misattributions of familiarity [1] and source monitoring errors [12]. These advances allow us to develop precise specifications of the conditions and processes that lead to a wide range of human memory errors.

Section 2 considers the characteristics of the ATC domain that are modelled in this paper, and the operator error based hazards which we provide formal definitions of. In section 3, we give a formal Z [17] model of an air-traffic control system. The Z model captures the state of the system, and the allowable operations that a controller may engage in. Section 4 formally models scenarios that require controller intervention (i.e., that are precursors to the occurrence of undesirable states, such as loss of separation). In section 5 we consider the significance of the work, and our future plans for the project.

2 Hazards in the Air-Traffic Control domain

We model an en-route sector of airspace. For example, such a sector might look like that depicted in Figure 1. The model presented in this paper considers only movement of aircraft through the sector in a two dimensional plane. Coordinates in the model are of the form (*latitude*, *longitude*) where the latitude and longitude are relative to the sector (and so are not necessarily global coordinates). For example, in Figure 1 aircraft 123 is at latitude 129 north and longitude 33 east. Vertical separation of aircraft is not considered here.

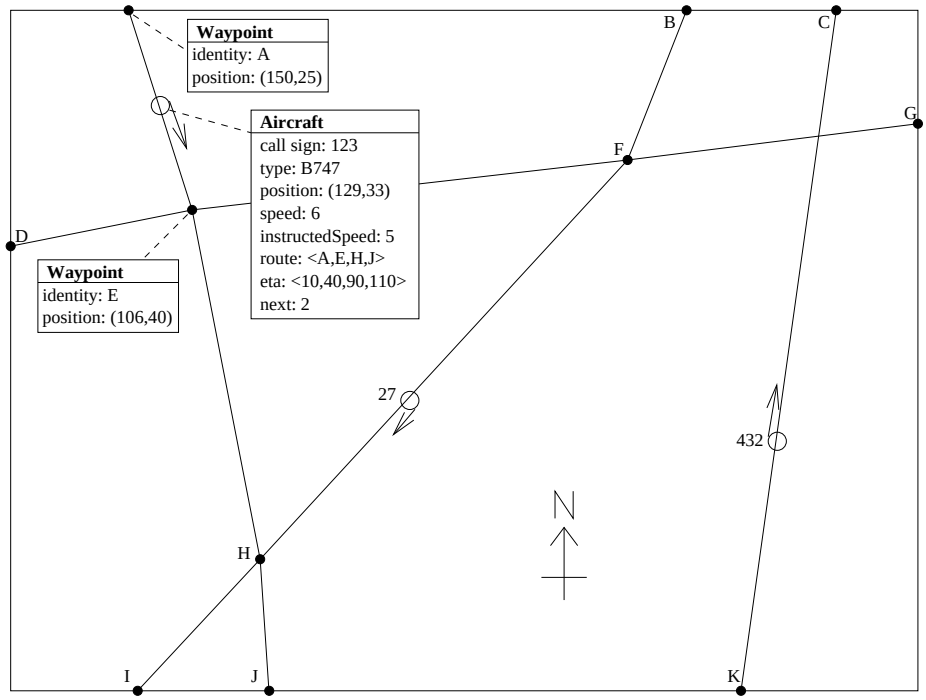


Figure 1: An en-route sector of controlled airspace

The focus of this model is the analysis of controller errors in managing a sector of air space. Consequently, aspects of the model relating to pilot behaviour are simplified. It is assumed, for example, that aircraft follow the flight paths meticulously. In this sense the model more closely reflects a simplistic computer simulation of an air sector than it reflects reality.

The primary task of controllers, in managing a sector of airspace is to ensure that the aircraft moving through the sector remain separated by a defined "minimum separation" distance. This distance may be 5 nautical miles for example. Controllers are also concerned with ensuring that aircraft move efficiently through the sector, with minimal delay and disruption, and that the movement of aircraft is orderly (e.g., aircraft are evenly spaced as they approach concentration points). For the purpose of this paper, however, we are concerned only with the safety requirement that minimum separation be observed.

In this paper we consider two scenarios in which operator error is possible, and the occurrence of such error has the potential to contribute to a loss of aircraft separation:

1. Convergence, when two aircraft are each travelling on two separate routes that merge. This situation is depicted in Figure 2.

2. Overtaking, when two aircraft are travelling on the same route in the same direction, and

the leading aircraft is slower than the following aircraft. This situation is depicted in Figure 3.

Both of these scenarios involve the aircraft reaching the culminating position close to the same time.

In the ATC domain, loss of separation is regarded as an accident situation. Hence both convergence and overtaking define hazardous system states in which the operator may need to intervene to ensure system safety. Failure to intervene, or an incorrect intervention, may be the initial event in an accident sequence that results in aircraft collision.

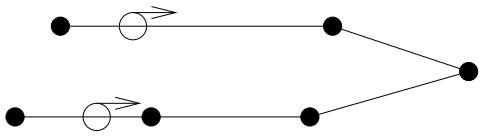


Figure 2: Convergence hazard

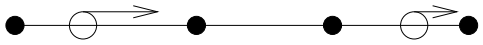


Figure 3: Overtaking hazard

The specification of the ATC system follows, then the formal definition of the convergence and overtaking hazards. Our model of the ATC system and hazards is given in Z. Because the hazards are formally defined in terms of the formal ATC specification, it would be possible to automate detection of the hazard conditions for a given ATC system configuration.

3 ATC System Formal Specification

The ATC system is a real-time system which includes various temporal aspects (specifically the estimated times of arrival of an aircraft at the waypoints in it's flight route). We define a discrete abstraction of time and provide a function for calculating the difference between two times.

$$Time == \mathbb{N}$$

$$abs : \mathbb{Z} \rightarrow \mathbb{N}$$

$$timeDifference : Time \times Time \rightarrow Time$$

$$\forall t1, t2 : Time \bullet \\ timeDifference(t1, t2) = abs(t1 - t2)$$

For simplicity we assume that there is a one-to-one correspondence between the ATC coordinate system and the pixel grid utilised by the system HCI. We define types for describing positions in this coordinate system.

$Latitude == \mathbb{N}$
$Longitude == \mathbb{N}$
$Position == Latitude \times Longitude$

For the ATC system HCI aircraft movement is assumed to follow a continuous, connected path between two positions, moving only horizontally or vertically in each step along that path. This movement of an aircraft on the system HCI is illustrated in Figure 4.

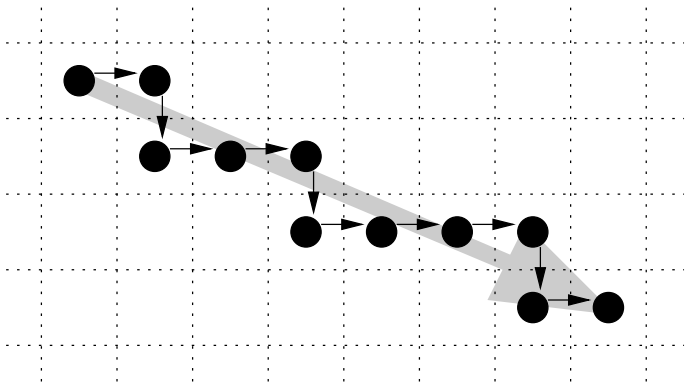


Figure 4: The movement of an aircraft across the pixel grid of the ATC system HCI.

The total distance travelled in this way by an aircraft is the Manhattan distance between the positions. The *distanceBetween* calculates this distance between two points.

$$distanceBetween : Position \times Position \rightarrow \mathbb{N}$$

$$\forall pos1, pos2 : Position \bullet distanceBetween(pos1, pos2) = abs(first(pos1) - first(pos2)) + abs(second(pos1) - second(pos2))$$

Aircraft in the ATC system move between waypoints, along defined routes.

$[WaypointID]$

$$\begin{array}{|l} Waypoint \\ \hline identity : WaypointID \\ position : Position \\ \hline \end{array}$$

$Route == \text{seq}_1\ Waypoint$

The function *subRoute* extracts the sub-route between the two numbered waypoints from a route.

$$subRoute : Route \times \mathbb{N} \times \mathbb{N} \pfun Route$$

$$\forall route : Route;\ start, end : \mathbb{N} \mid \{start, end\} \subseteq \text{dom}\ route \bullet subRoute(route, start, end) = (start\ ..\ end) \rhd route$$

The function *routeLength* determines the distance between the first and last waypoint on the route according to the Manhattan distance method as described above.

$$routeLength : Route \rightarrow \mathbb{N}$$

$$routeLength(\langle\,\rangle) = 0$$
$$\forall\, wp1, wp2 : Waypoint;\ route : Route \bullet$$
$$\quad routeLength(\langle wp1\rangle) = 0 \land$$
$$\quad routeLength(\langle wp1, wp2\rangle \frown route) =$$
$$\qquad distanceBetween(wp1.position, wp2.position) +$$
$$\qquad routeLength(\langle wp2\rangle \frown route)$$

The ATC system deals primarily with aircraft and the operations available on them. Some additional types are provided to model the state of an aircraft.

Aircraft each have a unique identity (a call sign), and are of a particular type (in this case three types are included in the model: Boeing 747, Boeing 767, and Cessna).

$$[Callsign]$$
$$AircraftType ::= B747 \mid B767 \mid Cessna$$

We assume that (in flight) no aircraft can travel slower than some defined minimum speed, and similarly no aircraft can travel faster than some defined maximum speed.

$$minimumSpeed, maximumSpeed : \mathbb{N}$$

Aircraft speed must be between this minimum and maximum speed.

$$Speed == minimumSpeed \,..\, maximumSpeed$$

The state of an aircraft in the ATC system is modelled by the following schema.

Aircraft

$$callsign : Callsign$$
$$type : AircraftType$$
$$position : Position$$
$$speed : Speed$$
$$instructedSpeed : \mathbb{F}\, Speed$$
$$route : Route$$
$$eta : \mathrm{seq}_1\, Time$$
$$next : \mathbb{N}$$

$$\#instructedSpeed \leq 1$$
$$\#route \geq 2$$
$$\#route = \#eta$$
$$\forall\, num : 1 \,..\, \#eta - 1 \bullet$$
$$\quad eta(num) < eta(num + 1)$$
$$next \in 2 \,..\, \#route$$

Each aircraft has a call sign (*callsign*), type (*type*), current position (*position*), current speed (*speed*), the speed that the aircraft must change its current speed to match (*instructedSpeed*), flight route (*route*), estimated times of arrival for the waypoints in the flight route (*eta*), and the next waypoint in the flight route to which the aircraft is heading (*next*). The estimated

times or arrival are assumed to be derived internally by the ATC system (and are not detailed in this model of the HCI).

An aircraft can be instructed to change its speed by storing the target speed in the set *instructedSpeed*. The following operation schema *changeSpeed* provides this means for instructing an aircraft to change its speed.

changeSpeed

$$\Delta Aircraft$$
$$target? : Speed$$

$$instructedSpeed' = \{target?\}$$
$$callsign' = callsign$$
$$type' = type$$
$$position' = position$$
$$speed' = speed$$
$$route' = route$$
$$eta' = eta$$
$$next' = next$$

Aircraft can move one position unit on each clock tick (but may not appear to move on every clock tick). When an aircraft moves, its actual speed may also change by one speed unit (to become either faster or slower). The change in both the aircraft position and speed is defined non-deterministically to reflect that these aspects are outside the control of the ATC system (they are input to the ATC system, for example by a radar device). In this sense an aircraft need not have an obedient pilot, in which case hazardous situations resulting from failure to obey the controllers instructions (or from other causes such as aircraft failure) are captured within the model.

move

$$\Delta Aircraft$$

$$callsign' = callsign$$
$$type' = type$$
$$route' = route$$
$$(1 \,..\, next - 1) \upharpoonright eta' = (1 \,..\, next - 1) \upharpoonright eta$$
$$next' = if (position' = (route(next)).position \land next \neq \#route)$$
$$\quad then\ next + 1$$
$$\quad else\ next$$
$$distanceBetween(position, position') \in \{0, 1\}$$
$$speed - speed' \in \{-1, 0, 1\}$$

Aircraft can be rerouted by changing the sub-sequence of waypoints in its flight route after the next waypoint (it is not possible to change where an aircraft has already been). The estimated times of arrival for the new route must be calculated by the system internally (these calculations are not specified).

reroute

$\Delta Aircraft$
$newRoute? : \text{seq}\ Waypoint$

$subRoute(route, 1, next) = subRoute(newRoute?, 1, next)$
$route' = newRoute?$
$callsign' = callsign$
$type' = type$
$position' = position$
$speed' = speed$
$instructedSpeed' = instructedSpeed$
$next' = next$

We define a function that determines whether the destination of an aircraft has been reached, and therefore whether it should be removed from the sector.

$destinationReached : Aircraft \rightarrow \mathbb{B}$

$\forall\, ac : Aircraft \bullet destinationReached(ac) \Leftrightarrow$
$\quad ac.next = \#(ac.route) \land$
$\quad ac.position = (last(ac.route)).position$

Similarly we define a function that determines the distance from the aircraft to a particular waypoint in its flight route.

$distanceToWaypoint : Aircraft \times \mathbb{N} \nrightarrow \mathbb{N}$

$\forall\, ac : Aircraft;\ wpnum : \mathbb{N} \mid ac.next \leq wpnum \leq \#ac.route \bullet$
$\quad distanceToWaypoint(ac, wpnum) =$
$\quad\quad distanceBetween(ac.position, ((ac.route)(ac.next)).position) +$
$\quad\quad routeLength(subRoute(ac.route, ac.next, wpnum))$

The schema *Sector* defines the static structure of an en-route air sector as depicted for example in Figure 1. Such a sector consists of a set flight paths through the waypoints in the sector, and encompasses the rectangular airspace bounded by the defined latitudes and longitudes.

Sector

$flightPaths : Waypoint \leftrightarrow Waypoint$
$waypoints : \mathbb{F}\ Waypoint$
$north, south : Latitude$
$east, west : Longitude$
$handOverPoints : \mathbb{F}\ Waypoint$

$\text{id}(Waypoint) \cap flightPaths = \varnothing$
$\forall\, posn : \{wp : waypoints \bullet wp.position\} \bullet$
$\quad south \leq first\, posn \leq north \land$
$\quad west \leq second\, posn \leq east$
$handOverPoints = \{wp : waypoints \mid$
$\quad first(wp.position) \in \{north, south\} \lor$
$\quad second(wp.position) \in \{east, west\}\}$
$\#handOverPoints \geq 1$

The ATC system HCI screen and associated operations are defined next. The ATC screen captures the detailed state modelled by the *Sector* schema and the set of aircraft within that sector.

Screen

$$
\begin{array}{l}
sector : Sector \\
aircraft : \mathbb{F}\, Aircraft \\
\hline
\forall\, ac1, ac2 : aircraft \bullet ac1.callsign = ac2.callsign \Rightarrow ac1 = ac2 \\
\forall\, ac : aircraft \bullet \\
\quad sector.south \leq first(ac.position) \leq sector.north \land \\
\quad sector.west \leq second(ac.position) \leq sector.east \land \\
\quad (\forall\, wpnum : 1 \,..\, \#ac.route - 1 \bullet \\
\qquad ((ac.route)(wpnum), (ac.route)(wpnum + 1)) \in \\
\qquad\quad sector.flightPaths) \land \\
\quad head(ac.route) \in sector.handOverPoints \land \\
\quad last(ac.route) \in sector.handOverPoints
\end{array}
$$

Initially the sector has no aircraft – other details of the sector, such as its layout, are not described.

ScreenInit

$$
\begin{array}{l}
Screen \\
\hline
aircraft = \varnothing
\end{array}
$$

We define internal operations to enable aircraft to enter the screen and exit the screen. As this is an en-route sector aircraft may only enter and exit the screen at the hand over waypoints.

aircraftEnterScreen

$$
\begin{array}{l}
\Delta Screen \\
craft? : \mathbb{F}\, Aircraft \\
\hline
\{ac : craft? \bullet ac.callsign\} \cap \{ac : aircraft \bullet ac.callsign\} = \varnothing \\
\forall\, ac : craft? \bullet \\
\quad \mathrm{ran}(ac.route) \subseteq sector.waypoints \land \\
\quad ac.position = (head(ac.route)).position \land \\
\quad ac.next = 2 \\
aircraft' = aircraft \cup craft? \\
sector' = sector
\end{array}
$$

aircraftExitScreen

$$
\begin{array}{l}
\Delta Screen \\
craft! : \mathbb{F}\, Aircraft \\
\hline
craft! \subseteq aircraft \\
\forall\, ac : craft! \bullet destinationReached(ac) \\
aircraft' = aircraft \setminus craft! \\
sector' = sector
\end{array}
$$

Similarly we define the internal operation that allows all of the aircraft in the sector to move according to the *move* operation schema.

aircraftMove

$$\Delta Screen$$

$$aircraft' = \{move \mid \theta Aircraft \in aircraft \bullet \theta Aircraft'\}$$

All controller interactions with the screen involve selection of the aircraft which is to be acted on (either by rerouting or by instructing a change in speed).

selectAircraft

$$\Delta Aircraft$$
$$\Delta Screen$$
$$craft? : Aircraft$$

$$craft? \in aircraft$$
$$craft? = \theta Aircraft$$
$$\exists\, craft : Aircraft \bullet$$
$$\quad craft = \theta Aircraft' \land$$
$$\quad aircraft' = aircraft \setminus \{craft?\} \cup \{craft\}$$

The *selectAircraft* operation is an auxiliary operation used in conjunction with either the *reroute* or *changeSpeed* operation to interactively reroute or instruct a speed change for an aircraft on the screen respectively. These are the operations actually available to the controller for interacting with the ATC system via its graphical user-interface.

$$ChangeAircraftRoute \mathrel{\widehat=} selectAircraft \land reroute$$
$$ChangeAircraftSpeed \mathrel{\widehat=} selectAircraft \land changeSpeed$$

The following is the periodical, non-interactive, internal screen action (acting on all aircraft on the screen) which performs all of the internal operations on aircraft on each clock tick.

$$updateScreen \mathrel{\widehat=} aircraftExitScreen \fatsemi aircraftMove \fatsemi aircraftEnterScreen$$

The real-time aspects of the ATC system are modelled in the environment of the ATC system. This environment includes the ATC screen itself, and the real-time clock.

ATCEnv

$$screen : Screen$$
$$clock : Time$$

A clock tick moves time forward one time unit.

clockTick

$$\Delta ATCEnv$$

$$clock' = clock + 1$$

Each clock tick is associated with an update of the ATC system according to the *updateScreen* schema.

$$tick \mathrel{\widehat=} clockTick \land \left[\Delta ATCEnv;\ updateScreen \,\middle|\, screen = \theta Screen \land screen' = \theta Screen'\right]$$

4 Formal Definition of Error Producing Conditions

As noted in the introduction, existing models of human error do not adequately describe conditions leading to error or the mechanisms for error. Current psychological theories on human error focus on the role of memory in the decision making process [13]. A central part of this process with respect to the ATC domain is the identification of hazardous scenarios by the controller. In this section we formally define simple models of the identification of the two hazardous sequencing scenarios discussed in Section 2: convergence and overtaking.

Modelling the identification of these hazardous scenarios involves two aspects. First, each hazardous scenario is characterised by some definable structure concerning the relative relationships of the aircraft in the scenario to each other. Secondly, controllers only identify aircraft in these scenarios as being a potentially hazardous scenario when certain cognitive conditions are also satisfied in relation to that structure. Specifically, there is the requirement that the controller foresees the participating aircraft reaching some culminating position at about the same time.

The inaccuracy in the controller capability to predict when the involved aircraft will arrive at the culminating position is captured in the formal model using the constant $separationTime$ defined as follows.

$$separationTime : Time$$

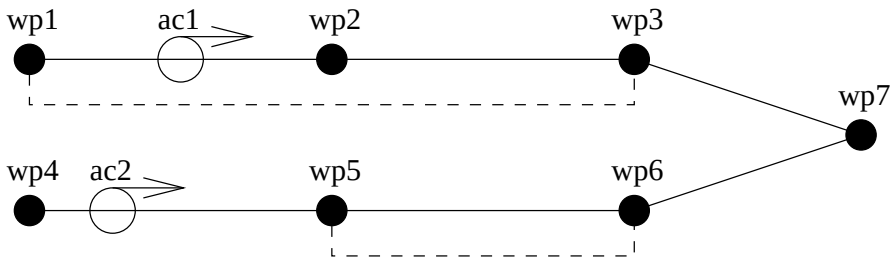
The controller is able to identify those hazardous scenarios in which the ETAs of the involved aircraft at the culminating position are within the $separationTime$ of each other. Practical experimentation is required to calibrate the value of $separationTime$.

The definitions that follow attempt only to model the cognitive identification of potentially hazardous scenarios in the ATC. Specifically, the definitions do not attempt to identify whether the controller perceives the hazard as likely (in which case intervening action is required) or unlikely (in which case intervening action is not required). Nor do the definitions attempt to assign the cognitive priority that the controller gives to each identified scenario.

In the ATC system, a hazardous scenario is identified by the set of aircraft participating in that scenario. Our formally defined functions for identifying hazardous scenarios are therefore modelled using the following type.

$$ScenarioIdentifier == (\mathbb{F}\, Aircraft) \rightarrow \mathbb{B}$$

A convergence hazard is characterised by the following situation:



In this situation the following waypoints are identified:

- $wp1/wp4$ - the previous waypoint passed by $ac1/ac2$

- $wp2/wp5$ - the next waypoint to be passed by $ac1/ac2$
- $wp3/wp6$ - the last waypoint before the routes of $ac1/ac2$ converge
- $wp7$ - the waypoint on which the routes of $ac1/ac2$ converge

This situation is identified as a convergence hazard if the following conditions hold:

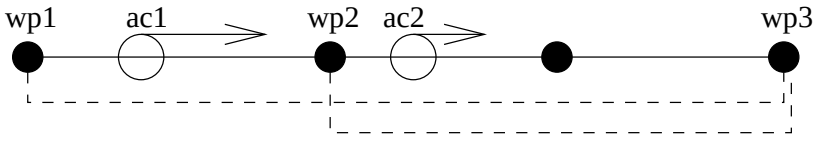
1. the sub-route of $ac1$ from $wp1$ to $wp3$ and the sub-route of $ac2$ from $wp5$ to $wp6$ include **no** common waypoints (that is $wp7$ is the *next* convergence of the two routes);
2. $wp3$ is distinct from $wp6$;
3. the time difference in the ETAs of the two aircraft at $wp7$ is less than the separation time.

The first of these conditions explicitly focuses on the next convergence in the routes of the two aircraft - previous convergences in the two routes (up to and including the waypoints $wp1/wp4$) are irrelevant. The second of these conditions applies to the scenario in which both aircraft are approaching the convergence waypoint $wp7$ - this specific scenario is otherwise not detected by the first rule.

Identification of convergence hazards is specified for the general case in the *convergenceHazard* function.

$$
\begin{array}{|l}
convergenceHazard : ScenarioIdentifier \\
\hline
\forall\, ac1, ac2 : Aircraft \mid ac1 \neq ac2 \bullet convergenceHazard(\{ac1, ac2\}) \Leftrightarrow \\
\quad (\exists\, point1 : ac1.next \,..\, \#ac1.route;\ point2 : ac2.next \,..\, \#ac2.route \bullet \\
\qquad (ac1.route)(point1) = (ac2.route)(point2) \land \\
\qquad (ac1.route)(point1 - 1) \neq (ac2.route)(point2 - 1) \land \\
\qquad \mathrm{ran}(subRoute(ac1.route, ac1.next - 1, point1 - 1)) \cap \\
\qquad\quad \mathrm{ran}(subRoute(ac2.route, ac2.next, point2 - 1)) = \varnothing \land \\
\qquad timeDifference((ac1.eta)(point1), (ac2.eta)(point2)) \leq \\
\qquad\quad separationTime)
\end{array}
$$

A overtaking hazard is characterised by the following situation:



In this situation the following waypoints are identified:

- $wp1$ - the previous waypoint passed by $ac1$
- $wp2$ - the previous waypoint passed by $ac2$
- $wp3$ - some later, common waypoint on the routes of $ac1$ and $ac2$

This situation is identified as an overtaking hazard if the following conditions hold:

1. the sub-route of $ac1$ from $wp1$ to $wp3$ includes the sub-route of $ac2$ from $wp2$ to $wp3$ (that is, $ac1$ and $ac2$ travel a common route from $wp2$ to $wp3$);

2. $ac1$ is further away from $wp3$ than $ac2$ (thus $ac1$ is initially following $ac2$ along the common route);

3. the ETA of $ac1$ at $wp3$ is before the ETA of $ac2$ at $wp3$, or
the ETAs of $ac1$ and $ac2$ at $wp3$ are within the separation time of each other.

Identification of this situation is specified in the *overtakingHazard* function.

$$
\begin{array}{|l}
overtakingHazard : ScenarioIdentifier \\
\hline
\forall\, ac1, ac2 : Aircraft \mid ac1 \neq ac2 \bullet overtakingHazard(\{ac1, ac2\}) \Leftrightarrow \\
\quad (\exists\, point1 : ac1.next \,..\, \#ac1.route;\ point2 : ac2.next \,..\, \#ac2.route \bullet \\
\qquad subRoute(ac2.route, ac2.next - 1, point2) \,\mathrm{suffix} \\
\qquad\qquad subRoute(ac1.route, ac1.next - 1, point1) \wedge \\
\qquad distanceToWaypoint(ac1, point1) > distanceToWaypoint(ac2, point2) \wedge \\
\qquad ((ac1.eta)(point1) < (ac2.eta)(point2) \vee \\
\qquad timeDifference((ac1.eta)(point1), (ac2.eta)(point2)) \leq separationTime))
\end{array}
$$

Where the controller does not take action in response to these first two hazards, a violation of the minimum separation distance between aircraft (defined by a regulatory authority) is likely to occur. Violation of minimum separation constitutes a mishap in the system. The minimum separation distance between aircraft and the identification of separation mishaps are defined as follows.

$$
\begin{array}{|l}
minimumSeparation : \mathbb{N}
\end{array}
$$

$$
\begin{array}{|l}
separationMishap : ScenarioIdentifier \\
\hline
\forall\, participants : \mathbb{F}\, Aircraft \bullet \\
\quad separationMishap(participants) \Leftrightarrow (\forall\, ac1, ac2 : participants \mid \\
\qquad ac1 \neq ac2 \bullet \\
\qquad distanceBetween(ac1.position, ac2.position) < minimumSeparation)
\end{array}
$$

We have formally identified two situations in which a mishap may potentially arise if the controller fails to intervene. The next phase of our project will investigate metrics for determining the likelihood of such errors given knowledge of the system configuration (e.g., number of aircraft on the screen, layout of the sector, etc.). Such metrics will be based on understanding of measures of controller effectiveness such as workload.

5 Conclusions

We have shown how formal models of hazardous situations sensitive to operator error can be produced using the Z specification language. The formally defined hazard identifiers *convergenceHazard* and *overtakingHazard* identify system states where cognitive failures such as contextual interference and prospective memory failures may result in an accident occurring. They indicate situations in which the controller should act, and where inaction or the wrong

action may result in loss of aircraft separation and potentially loss of life. In this paper, the particular domain for which the techniques have been demonstrated is air-traffic control, but we believe the method can be generalised to other safety-critical domains.

An important feature of the SafeHCI project is that it incorporates a model of the operator's cognitive state. This model identifies the information that the controller focuses on. In psychological terms, this model describes the information that may be represented in a controller's "working memory".

The development of a model of how controllers represent air traffic situations in working memory is an essential first step in the development of a model of human error within this task. The next step involves modelling the way that information is stored in long term memory, and the types of cues that are used to retrieve this information. We assume that each situation that a controller is exposed to is stored in long term memory, together with the contextual details associated with that situation. For example, if a controller sees an example of a conflict between two aircraft occurring on the approach to a particular airport, then she is likely to store information about the attributes of the aircraft involved (e.g., speed, position, altitude, type), the context (e.g., the fact that it occurred on the northern approach to Sydney during the morning rush hour), and the solution adopted (e.g., to slow aircraft A down when it passes a specific point). When the controller sees another pair of aircraft converging at the same location, then this may cue the retrieval of the earlier example from memory, together with the solution. The ability to retrieve past examples from memory is an important aspect of expertise, because it allows controllers recognise repeated conflict patterns, and draw upon prior solutions.

By understanding the cognitive mechanisms involved, we can make predictions regarding the conditions under which error is likely to occur. For example, errors can be caused by contextual interference. If a controller sees a problem out of context, then their ability to recognise the conflict and retrieve the correct solution will be impaired. Alternatively, if a controller sees a new problem that is superficially similar to previous conflicts that she has seen in a particular context, then she may retrieve the wrong solution memory.

Later steps in the project will produce formal models of the cognitive processes involved in identifying and resolving ATC hazards and will validate the modelling notation and analysis method through empirical studies. The results will show whether the mechanisms that have been identified really do produce errors. The outcome of the experiments may motivate improvements and changes to the notation and method. Later in the project, an industrial case study will be conducted, in which the notation and method will be used to analyse an industrial system.

The project opens up several possibilities for further work:

1. Extension and generalisation of the method to enable modelling and analysis of a broad range of safety-critical interactive systems with respect to a wide range of error types.

2. Extending the analysis method to provide support for identification of corrective measures using pattern-based techniques, e.g., as described in work carried out by one of the chief investigators [10].

3. Provision of tool support for the method, including tools for producing appropriate formal specifications and for performing and recording analyses.

4. Further experimental validation of the method in collaboration with industrial partners.

5. Extension of the scope of the approach to include more detailed aspects of presentation.

References

[1] B. Hesketh A. Neal and S. Andrews. Instance-based categorisation: Intentional versus automatic forms of retrieval. *Memory and Cognition*, 23:227–242, 1995.

[2] Commonwealth of Australia. Australian Defence Standard DEF(AUST) 5679: The Procurement of Computer-based Safety Critical Systems. Department of Defence, 1998.

[3] R. Duke, G. Rose, and G. Smith. Object-Z: A Specification Language Advocated for the Description of Standards. *Computer Standards and Interfaces*, 17:511–533, 1995.

[4] G. O. Einstein and M. A. McDanial. Normal aging and prospective memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 16:717–726, 1990.

[5] European Committee for Electrotechnical Standardization. European Standard prEN 50128: Railway applications; Software for railway control and protection systems. CENELEC, 1995.

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

[7] A. Hussey. Safety Analysis of User-Interfaces at Multiple Levels of Interaction. In P. Lindsay, editor, *3rd Australian Workshop on Industrial Experience with Safety Critical Systems and Software*, pages 41–57. ACS, 1998.

[8] A. Hussey and D. Carrington. Which widgets? Deriving Implementations from Formal User-Interface Specifications. In P. Markopoulos and P. Johnson, editors, *DSV-IS '98*, pages 239–257. Springer-Verlag, 1998.

[9] International Electrotechnical Commission. 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems. IEC, 1997.

[10] B. Kirwan, editor. *A Guide to Practical Human Reliability Assessment.* Taylor and Francis, 1994.

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

[12] S. Hashtroudi M. K. Johnson and D. S. Lindsay. Source monitoring. *Psychological Bulletin*, 114:3–28, 1993.

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

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

[15] F. Paterno. A Formal Approach to the Evaluation of Interactive Systems. *SIGCHI Bulletin*, 26(2):69–73, April 1994.

[16] J. W. Senders and N. P. Moray, editors. *Human Error: Cause, Prediction and Reduction.* Lawrence Erlbaum Associates, 1991.

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