ELECTRONIC **WORKSHOPS IN COMPUTING**

Series edited by Professor C.J. van Rijsbergen

C.R. Roast and J.I. Siddiqi, Sheffield Hallam University, UK (Eds)

BCS-FACS Workshop on Formal Aspects of the Human Computer Interface

Proceedings of the BCS-FACS Workshop on Formal Aspects of the Human Computer Interface, Sheffield Hallam University, 10-12 September 1996

The User Context and Formal Specification in Interactive System Design

M. Harrison, R. Fields and P.C. Wright

Published in Collaboration with the British Computer Society

BCS FACS

Springer

ISBN 3-540-76105-5

The user context and formal specification in interactive system design

M. D. Harrison, R. Fields and P.C. Wright
Human Computer Interaction Group,
Department of Computer Science, University of York,
Heslington, York, Y01 5DD, U.K.
email: {mdh,bob,pcw}@minster.york.ac.uk

Abstract

The growing use of computer-based systems in many sectors, particularly those where safety is an issue, combined with the increased complexity of human interaction with such systems, has meant that concerns of human error and usability in general are more important than ever. This paper reviews research done at York to relate formal methods to Human Computer Interaction. A particular concern of this work has been to show how understandings of use can be integrated into the design of the system. These techniques have evolved to take increasing account of the user context in the specification of the system. The paper will illustrate this development and show how formal specification may be used as a common representational framework for user's goals, tasks and for components of the interface specification. The method is illustrated using a small example from the aircraft domain.

1 Introduction

The relevance of formal specification to the development of large systems continues to be questioned even as these techniques have matured (see for example the special issue of IEEE Computer [29]). There is argument for example that the primary motivation for their development, the correctness problem, is less significant than envisaged.

For example, MacKenzie [24] claims that, during the period of his study, while software error in safety critical systems contributed to 3 % of approximately 1100 computer related accidental deaths, 92 % were concerned with human factors. Since similar findings are reported by other authors, for example [18, 30], it seems clear that a significant improvement in safety might be achieved if the possibility and consequences of human error can be reduced through design. Formal methods have a role here if they can facilitate analysis of unforeseen consequences in interaction and if they can support systematic analysis based on expected behaviour.

In 1991, a paper about human error tolerant systems [15] made some initial proposals about how psychological understandings of human error might be used as a basis for formulating properties of human error resilience in interactive systems. This paper shall be used as a starting point here. The deficiencies of attempts to comprehend human centred theories in the design of interactive systems through formal properties shall be illustrated, and we shall show how our work has evolved to take increasing notice of the work context. We shall discuss the role of experiment in validating principles and how a notion of template might be used to encapsulate assumptions about how the system is being used. In this connection an iterative design method has been explored based on modelling, prototyping and evaluation. This work was limited to a single user of a system and to office systems such as hypertext and simple databases [14].

This empirical approach made no assumption about context of use and was based on laboratory style experiments. Over the last few years HCI in general has become much more concerned about the role of the wider context, the co-operative work system and the organisation. Papers and books about the deficiencies of traditional approaches to human computer interaction have become popular, see for example the special theme of the Communications of the ACM [31], and books about activity theory [26] and distributed cognition [19]. We shall describe how our work has taken a task focus to attempt to alleviate this deficiency. We shall discuss the limitations of a simple task focus and establish requirements for richer analyses of context in design.

In summary, throughout the development of these techniques two questions have been important:

- what is an appropriate method of specifying interactive systems so that proper priorities about use may be established?
- how can systems be designed taking into account a proper understanding of the broader work context?

2 Requirements Specification

We, see for example [15], have proposed that general principles be applied in the design of interactive systems to achieve a measure of human error resilience. Three principles: visibility, predictability and recoverability were considered in particular.

Visibility is concerned with whether relevant state of the system is perceivable. Visibility requires that all the “key” state attributes are perceivable at an appropriate level of detail, for example the text of the document. Visibility is extended to include the possibility that a larger state than could be perceived immediately is accessible through browsing actions (*observability*).

Predictability is concerned with the possibility of predicting the effect of a command on the basis of what is perceivable about the system. Mistakes then can be protected against by providing a perceivable model of how the system works. This notion of predictability is akin to observational equivalence (CCS [25]) and is in general undecideable.

Recoverability, is concerned with two ideas. One notion is concerned with whether the user is able to recover from the current state of the system to an earlier state. Hence it should be possible to recover from mistakes. The second notion is about whether the user can continue to achieve the same goal after a mistake. Abowd and Dix analyse properties in the first category in the context of group editing [1]. Harrison and others consider properties of the second category [16].

Formulations of these principles are abstract and use generic stimulus response style models of interactive behaviour. The advantage of generality is at the cost of applicability. The PIE model [6] allows discussion of predictability but it is not clear how a system might be constructed that is predictable.

More detailed principles may be specified that concern the resources and capacities of the user. For example, the principle that immediately relevant commands should be directly accessible in the current mode is particularly relevant to the design of complex dynamic systems such as aircraft cockpits. A systematic approach to analysing what is available in a given mode or mission phase is already an essential component of the design of such systems. Procedures to check such principles are already carried out using informal inspections in some sectors of industry. Consistency is another system property that supports appropriate model generalization and thus reduces the likelihood of error. It is a notion that should be used carefully because inappropriate or partial consistencies may have the effect of leading to inappropriate generalization (see [12]).

General principles such as these are problematic for a number of reasons.

- Producing a formalisation of the fact that information must be preserved in the relation between state and presentation is not easy. How does the specifier know whether the specification of the presentation of the information will actually make the information perceivable to the user? For example, how should time be presented? At what level of granularity? Relevant for what tasks?
- It is not usually appropriate to make principles general across the whole interactive system. Decisions have to be made about where the properties should hold. State information will be relevant or essential to understanding how to invoke commands or tasks and this information should be made perceivable.

A view of the specification of the interactive system is required that will give coherence to principles that only apply to parts of systems. Emphasising the salience of aspects of the state, presentation or input can be achieved by using a notion of *template* to capture what information is required to perform a subset of actions or to describe a hypothesis about what the user would recognise in performing actions [28]. Templates were used in [14] as a basis for evaluating a design against assumptions about what the user would recognise as important in searching or editing a database. Templates may also be important to aid the designer create display layouts such as would be required to support a pilot's recovery from some failure using a standard operating procedure. Templates enhance the formal specification by adding components that are not used in the implementation of the system. For example they focus on parts of the implemented display that happen to be relevant to what the operator is doing, or they describe relations between data components (that one value is greater than another for example). They can also be used to restrict properties such as predictability, visibility or recoverability to state, presentation or input that is contained in a template.

Dearden and Harrison [5] discuss the role of abstract models of HCI and in particular a class of models, intermediate between black box models like the PIE model and more concrete models which are discussed later in the paper. The style of specification that has been introduced so far is used to emphasise the interactive aspect of the computer system. It can be used as a framework for representing general properties of interactive systems and for specifying hypotheses about what pieces of the display will be salient in the use of the system. The paradigm of design is top down, general, and based on the system as an artefact used by an individual. The problem with these general models is that, while useful if understood in context, they provide an impoverished understanding of the resources that the user brings to the system. They presume, tacitly, a laboratory environment in which the user and computer system co-operate. In practice, users are influenced by culture, they are completing other tasks, they are in a noisy environment. Two issues must be dealt with if usability and error resilience are to be taken seriously. More detailed specifications of the system and a richer understanding of context are required.

3 Interactor Specifications

An approach to specification is required that is compatible with standard techniques and will permit formulation and analysis of interactive aspects of the system. The technique should maintain the prominence of issues of user interaction and support checking and analysis. A class of specification approaches have been developed based on variants of standard techniques. What the approaches have in common is that they are based on an object oriented decomposition of system into simpler units, each of which is specified by describing its internal state, an indication of whether and how attributes of the state are made perceivable to the user and the state-changing operations that are available. Hence properties like *visibility* of information and *availability* of actions are intrinsic components of the system specification. The interface between an interactor and its environment involves the perceivable components of the state but also consists of a set of actions that can be triggered by external interactions either with users or with other interactors in the system. The primary unit of description in this model is referred to as the *interactor*. As an example, an interactor specifying the behaviour of a simple button can be defined as follows.

interactor *toggle-button*
attributes

$\boxed{vis}\, enabled : \mathbb{B}$

initially

$enabled = \mathsf{false}$

actions

press

operations

action *press*
pre true
post $enabled = \neg \overleftarrow{enabled}$

In most respects the specification has the attributes of a standard model oriented approach to specification. In the example being illustrated here VDM [20] is the basis. An interactor's *state* is represented as a collection of typed attributes or state variables. In this case, there is only one: the Boolean valued *enabled*, which is initially false. The enabled property of a button can be perceived visually, as indicated by the $\boxed{vis}$ annotation. Such perceivable variables are called percepts, and make up the presentation component of the interactor (we discuss different modalities in the presentation component in [8]). A single *action*, *press*, is available in the interface of the *toggle-button* interactor, and its effect is to invert the *enabled* state variable. This behaviour is described precisely by the pre- and post-conditions of the operation definition: if the operation is invoked in any state (indicated by the true pre-condition), then the effect, specified by the post-condition, is that *enabled* becomes the negation of its value prior to the operation's execution ($\overleftarrow{enabled}$). In the next section, when we discuss the context of the tasks for which a system will be used, operations' pre- and post-conditions will be used, not only to specify what the system should do, but also to help in investigating what effect certain courses of action on the part of the user have on a system's state.

An important feature of the interactor notation is that it supports a form of modularity where copies of interactors may be used in other interactors. Renaming of the included interactors allows multiple copies of the same entity to be distinguished (see [7] for more on interactor inclusion in the context of a similar system). The next example shows how the notation can be used to describe the behaviour of a slightly more complex system. The interactor *MessageDisplay* has two state variables: a queue of messages (the internal structure of which will not concern us here), and a visible optional *current* message. An invariant constrains the *current* message to be the first in the queue, or the special distinguished value nil if the queue is empty. This specification is a fragment of the description of a much larger system for handling air traffic control-related messages [13, 7].

The existence of a type *Message* is assumed. Messages are prioritised, and a function $pr: Message \to \mathbb{N}$ takes each message to its priority level. The third conjunct of the invariant insists that the order of messages in the queue reflects their priorities.

interactor *MessageDisplay*
attributes

$$queue : Message^*$$
$$\boxed{vis}\ current : [Message]$$

invariants

$queue = [\,] \ \Rightarrow\ current = \mathsf{nil}$
$queue \neq [\,] \ \Rightarrow\ current = \mathsf{hd}\ queue$
$\forall i, j \in \mathsf{inds}\ queue \cdot i \leq j \ \Rightarrow\ pr(queue[i]) \leq pr(queue[j])$

actions

clear

operations

action *clear*
ext wr *queue*, *current*
pre $queue \neq [\,]$
post $queue = \mathsf{tl}\ \overleftarrow{queue} \wedge$
$current = (\mathsf{if}\ queue = [\,]\ \mathsf{then}\ \mathsf{nil}\ \mathsf{else}\ \mathsf{hd}\ queue)$

action *receive* $(m: Message)$
ext wr *queue*, *current*
post $\exists l, r: Message^* \cdot (\overleftarrow{queue} = l \frown r) \wedge (queue = l \frown [m] \frown r) \wedge$
$(l = [\,] \vee pr(last(l)) \leq pr(m)) \wedge$
$(r = [\,] \vee pr(\mathsf{hd}\ r) > pr(m)) \wedge$
$current = \mathsf{hd}\ queue$

The state variables can be altered by two operations: *clear* which discards the current message, and *receive* which handles the arrival of a new message by adding it to the appropriate place in the queue. The *clear* operation is available to the user of the system as a user interface action, whereas *receive* is not available to the user and is used to manage the communication between *MessageDisplay* and other components of the message handling system (which could also be specified as interactors).

Why use formality?

In common with many other modelling activities, formal interactor specifications as used above, can be used in two roles: prescriptive and investigative. On the one hand, the specification allows the designer to capture requirements of how the system must behave, and assumptions about how much of this behaviour it must reveal to the user. On the other hand, the model allows the designer to explore the answers to a range of questions both by informal inspection, and by the use of formal techniques with a view to validating the specification.

For example, the designer might ask the question whether the user can gain access to all relevant parts of the system state. This question can be answered informally by inspecting the specification. Only one message *mesg* can be observed at a time, and the invariant guarantees that it is always the first message in the queue. In order to make other messages available, the queue itself must change. New messages of higher priority displace the first message; older messages can only be observed by removing the currently displayed message, and these actions cannot be undone.

This level of discussion may be adequate to assure the system developer of the scope of the possible implementation based on this specification. Properties may also be checked more formally using the same specification. For instance, the designer may want to be convinced that the receipt of a message may require that the displayed message changes; the property that if the current message has priority 2 and a priority 1 message arrives, then the new message becomes the displayed one, can be formalised as the following inference.

$$\frac{\begin{array}{c} m, mes, mes' : Message; q, q' : Message^* \\ pr(mes) = 2; pr(m) = 1; \textit{post-receive}(m, q, mes, q', mes') \end{array}}{mes' = m}$$

A rigorous proof of this property is not difficult and relies on the invariants, the definition of the *receive* operation and the basic inferences of the VDM specification language [2].

h1.	$m, mes, mes' : Message; q, q' : Message^*$	
h2.	$pr(mes) = 2; pr(m) = 1$	
h3.	$\textit{post-receive}(m, q, mes, q', mes')$	
1.	$\exists l, r : Message^* \cdot$	
	$q = l \frown r \wedge q' = l \frown [m] \frown r \wedge$	
	$(l = [\,] \vee pr(last(l)) \leq pr(m)) \wedge$	
	$(r = [\,] \vee pr(\mathsf{hd}\, r) > pr(m)) \wedge$	
	$mes' = \mathsf{hd}\, q'$	*receive* defn(h3)
2.	from $q = l \frown r \wedge q' = l \frown [m] \frown r \wedge$	
	$(l = [\,] \vee pr(last(l)) \leq pr(m)) \wedge$	
	$(r = [\,] \vee pr(\mathsf{hd}\, r) > pr(m)) \wedge$	
	$mes' = \mathsf{hd}\, q'$	
2.1	from $l = [\,]$	
	infer $mes' = m$	Lists(h2,h2.1)
2.2	from $pr(last(l)) \leq pr(m))$	
	infer $mes' = m$	Contradiction(h2, h2.2)
	infer $mes' = m$	$\vee$-elim(2.1, 2.2, h2
c.	$mes' = m$	$\exists$-elim (1,2)

Although this property is very simple and is more about the functional properties of the specified system than its user interface or usability, such properties can form the basis of more interesting proofs. For example, a question about the system as specified that relates directly to its effectiveness is whether messages are ever lost. In other words, will situations arise where messages arrive and are deleted by the user without having been read? A situation in which this could occur is if a message arrives and the user presses *clear* almost immediately afterwards (e.g., because they had read the previous message). In fact, the newly received message will no longer be present in the system, as demonstrated by a proof of the following inference (simplified by the removal of typing hypotheses):

$$\frac{\begin{array}{c} pr(mes) = 2; pr(m) = 1; \\ \textit{post-receive}(m, q, mes, q', mes'); \\ \textit{post-clear}(q', mes', q'', mes'') \end{array}}{m \notin \mathsf{elems}\, q''}$$

The proof of this property is straightforward and can make use of the previously proved result.

4 Tasks as part of Context

The interactor approach provides the means by which the specifier can be encouraged to consider issues of user interaction, and be provided with some power to enable analysis of the system. What it still does not do is force the designer to consider the broader work context in the design of the system. In practice, many errors that occur in systems are context conditioned. This point is made by Grudin [12] in his paper on the role of consistency in interactive systems. It is also illustrated by the case of Coordination Systems[1] [33] which force inappropriate formality in communication in office environments, but appear to be useful in airborne datalink systems, where time critical communications, such as would be required to co-ordinate navigation between pilots and air traffic control, require accuracy. It should be emphasised that analysis of improvement in the second case is a complicated matter. The introduction of datalink technology involves shifting responsibility and the usability problems may have simply moved somewhere else.

A method for incorporating task focus can be taken from the collection of techniques referred to as Human Reliability Assessment [21], which deal with the assessment of human error in interactive systems. The approach presumes some kind of system description, then presumes a task analysis which has the effect of giving some context to the analysis. The task analysis is used as a basis for analysing where "mutations" of assumed behaviour may occur because of human error. The idea then is to judge the likelihood and impact of human errors that might arise as a result of particular mutations. The interactor approach may be enriched by adopting techniques from Human Reliability Assessment.

An example

Consider as an example, the specification of a simple interface to an engine fire extinguisher on board the flightdeck of a twin-engined aeroplane. For each of the two engines (left and right) there are two fire extinguishers, and the extinguishers and other controls associated with an engine are accessed via an engine instrument panel. The behaviour of each instrument panel can be described by the following interactor:

[1] Structured tools for computer mediated conversation that impose message types on the communicators.

interactor *EnginePanel*
attributes

$$
\begin{aligned}
aircraft &: aircraft\_state \\
\boxed{vis}\ protected &: \mathbb{B} \\
\boxed{vis}\ avail_1, avail_2 &: \mathbb{B} \\
\boxed{vis}\ is\_fire &: \mathbb{B}
\end{aligned}
$$

initially

$(protected \wedge avail_1 \wedge avail_2 \wedge \neg is\_fire)$

actions

$release, protect, ext_1, ext_2$

operations

action *release*
pre *protected*
post $\neg protected$

action ext_1
pre $\neg protected \wedge avail_1$
post $\neg avail_1$

action *protect*
pre $\neg protected$
post *protected*

action ext_2
pre $\neg protected \wedge avail_2$
post $\neg avail_2$

action *detect_fire*
post *is_fire*

action *cancel_fire*
post $\neg is\_fire$

The availability status of each of the extinguishers, that is, whether or not they have been used up, is recorded by two state variables $avail_1$ and $avail_2$. The two fire extinguishers are activated by the actions ext_1 and ext_2. If an extinguisher is available and the extinguisher protection is off, then performing the corresponding *ext* action causes the extinguisher to become unavailable (i.e., used up). The *protected* state variable represents a feature common in safety critical interfaces: a *guard* which aims to make certain “high consequence” actions (such as turning off an engine or using up a fire extinguisher) harder to perform. The effect of having this guarding mechanism means that in order to operate the extinguisher effectively, the pilot must first carry out the *release* action to “turn off” the protection.

The two actions *detect_fire* and *cancel_fire* manipulate a variable recording the presence or otherwise of a fire. Although they are specified in the interactor, no causal connection is described between these operations and the operation of the extinguishers. This is appropriate because this system cannot define whether an attempt to extinguish a fire will be successful. The fire operations are added because part of the context argument will involve a discussion of the impact of reducing the likelihood of extinguishing a fire.

This simple interactor could be used to prove predictability and visibility properties. The effect of sequences of interactions may be explored to ensure that appropriate state of the extinguisher is always visible or to ensure that extinguishing actions are always predictable. However a clearer understanding of the context in which this system is used is required.

Tasks

A significant step towards understanding the context in which a system will be used is to gain an insight into what the user should do to achieve particular goals (sometimes referred to as the Designers’ Intended Procedure). In order to do this, many task analysis techniques such as Hierarchical Task Analysis or HTA (see [22]) have been developed. Although originally intended to capture the knowledge humans have about the tasks they carry out, HTA can be used to represent a designer’s or analyst’s assumptions about a user’s *goals* in performing some interaction, the decomposition of goals into *sub-goals*, and *plans* that describe how the sub-goals are composed to achieve the higher-level goal. Task analysis allows descriptions to be made or hypotheses formed about the knowledge an expert has of the system, and of the work to be done. The aim here in using task analysis is both to provide input to the design process and to present a convenient starting point for the assessing the system's response to error.

Since one idea underlying this type of task analysis is that goals are decomposed into a collection of sub-goals and a plan stating the order in which the sub-goals are satisfied (and any iteration, choice, and so on), the CSP process algebra notation [17] may be used to describe constraints on the sequence in which sub-goals are carried out. The purpose of the formal description is to be precise about what the user needs to know with the aim of being more systematic about the potential for deviation from normal behaviour. CSP is chosen because it is a standard approach and because the notation supports rules for generating traces.

The subset of CSP used here allows actions and sub-goals to be named and allows the description of sequential (; and $\longrightarrow$) or parallel ($\|$) execution of sub-tasks, choice between alternative courses of action ($\Box$) and recursion or repetition (μ).

In the case of the engine fire extinguisher, the task being considered is likely to be prescribed as an operating procedure that must be adhered to by the pilot. The fire extinguisher panel interactor is used to refer to the left engine by appending "Left" to each action name. The task involves some of the actions of the fire extinguisher but also includes other actions associated with identifying where the fire is and actions of waiting (the pilot doing nothing), idling and closing the engine. In this example most of the actions are carried out sequentially. There are choice points however concerned with deciding whether the fire extinguisher has been effective.

$$
\begin{aligned}
&\mathit{engFireLeft} = \\
&\qquad \mathit{identifyEngLeft}; \mathit{releaseLeft}; \mathit{idleEngLeft}; \mathit{wait}; \mathit{ext}_1\mathit{Left}; \mathit{checkEngLeft}; \\
&\qquad \mathit{engOK} \longrightarrow \mathit{closeEngLeft} \,\Box \\
&\qquad \mathit{engNotOk} \longrightarrow \mathit{ext}_2\mathit{Left}; \mathit{closeEngLeft}
\end{aligned}
$$

It should be noted that actions have been presumed which are at an appropriate level of abstraction for this particular analysis. In practice, the action *closeEngLeft* could be decomposed into lower level actions that the pilot must carry out in order to close the engine. The CSP descriptions here play a role similar to that of the plan components of hierarchical task analysis by describing constraints on the order in which sub-goals and actions are combined to achieve some goal. CSP provides rules for generating traces from process descriptions. Hence if the task is correctly carried out the following traces (and their prefixes) would be observed.

$$
\begin{aligned}
&\langle \mathit{identifyEngLeft}, \mathit{releaseLeft}, \mathit{idleEngLeft}, \mathit{wait}, \mathit{ext}_1\mathit{Left}, \mathit{checkEngLeft}, \mathit{engOk}, \mathit{closeEngLeft} \rangle \\
&\langle \mathit{identifyEngLeft}, \mathit{releaseLeft}, \mathit{idleEngLeft}, \mathit{wait}, \mathit{ext}_1\mathit{Left}, \mathit{checkEngLeft}, \mathit{engNotOk}, \mathit{ext}_2\mathit{Left}, \mathit{closeEngLeft} \rangle
\end{aligned}
$$

This description of traces can be used to provide a basis for identifying possible erroneous deviations from normal behaviour. Guidewords, words that are used as reminders for the analyst, may be used. For example, guidewords based on the phenotypes of [18], can be used systematically to identify the effect of mutations that might result from the user *omitting* actions; *inserting* actions inappropriately; *reversing* pairs of actions; *repeating* actions. The problem is that an exhaustive analysis such as this can generate a very large number of possibilities. Two methods can be used to restrict the set of possibilities. Firstly, any constraints embedded in the interface will have the effect of reducing the space of error traces that are actually possible, thereby reducing the size of any analysis. Indeed, such interface constraints are often built into systems with the express purpose of reducing the potential for error (see, for example Norman's discussion of interlocks and similar devices [27]). A constraint which is likely to exist in our fire extinguisher example is that the protection mechanism actually prevents the *ext* actions taking place. In CSP this constraint can be expressed as as a process:

$$
\mathit{System} = \mu X.(\mathit{releaseLeft} \longrightarrow \mu Y.(\mathit{ext}_1\mathit{Left}; Y \,\Box\, \mathit{ext}_2\mathit{Left}; Y \,\Box\, \mathit{closeEngLeft}; Y \,\Box\, \mathit{protectLeft}; X) \,\Box\, \mathit{closeEngLeft}; X)
$$

CSP includes rules for combining processes [17]. These can be used to produce a subset of possible trace mutations of the task description. For example, a trace that could potentially be generated by using the rules for mutation systematically would be one in which the *releaseLeft* action is omitted. The interaction between this mutated *engFireLeft* trace and system described above produces no valid trace. On the other hand, any trace is permitted that involves *wait* at any stage in the interaction — no constraint is imposed on the wait — and therefore *wait* and $\mathit{ext}_{\{1,2\}}$ actions might be reversed. A system can be made more human error resilient by forcing interaction to a more limited set of traces thereby guarding against certain kinds of error. We may say that we are reducing the *interaction potential* of the system (see also [16]). There are advantages and disadvantages of constraining behaviour. Well publicised examples of where constraint has not quite worked can be found in some envelope protection mechanisms used in fly-by-wire aircraft.

A second mechanism for reducing the set of mutations that must be considered is also suggested in HRA techniques. Further understanding of possible mutations is achieved by combining the external model of interaction mutation based on phenotypes with a deeper understanding of some of the cognitive processes involved in interacting with the system. The cognitive model that is used in, for example [10], is Norman's control loop. Here it is assumed that certain types of error are more likely to occur in the goal formulation phase of the cycle in a particular interaction than say the perception phase. This understanding of phases is combined with an understanding of what the user has to do in the interaction.

Having reduced the possible set of mutations by this means the next stage is to judge the significance or impact of some of the interactions. In the description of interactors the actions that are described at the lowest level of the task description are described as pre- and post-conditions; recall the definition of the ext_1 action:

action ext_1
pre $\neg protected \wedge avail_1$
post $\neg\, avail_1$

which describes the action of firing the first of the fire extinguishers. In the same way, see also [11], tasks may be expressed not only in terms of the sequences of actions but also in terms of the pre-condition that must be satisfied and the post condition that is achieved:

goal $engFireLeft$
pre $protected \wedge is\_fire$
post $\neg\, is\_fire$

The notions of goal and plan can thus be described formally as *state transformations* and *processes* in the same VDM [20] and CSP [17] notations that are used to describe interactive systems. As each goal is decomposed, new features may be added to the system to support the decomposition; in particular, new state variables may be required to allow the lower-level goals to be specified. The process of goal decomposition may continue until the goals may be implemented directly as system actions.

Given the definition of the goal and the task and the lower level description of the pre- and post-conditions for the actions it then becomes possible to judge the effect of the mutated traces that have been identified through the process described above. Part of this process can be carried out by systematically matching pre- and post-conditions of the individual actions and calculating what the effect of a mutation will be.

However this can only be part of the story. As already noted, the task descriptions also contain actions that are not contained in the descriptions of the interactor. For example no matter whether or not there is actually a fire the same sequence of actions will achieve the same result of firing extinguisher 1. The effect of the two traces (restricted only to actions of the interactor) that can arise can be analysed: $\langle release, ext_1, ext_2, protect \rangle$ on two states, the first in which is_fire is true and the second in which is_fire is false. Note that the trace in both cases is possible even though the effect of the trace when there is no fire is to waste the extinguisher. In [4] a number of notions of equivalence are provided that could be used in the automation of the process of checking for such situations.

By this means the system specification can aid the provision of a richer account of task context. However this notion of context in the design of interactive systems is still not rich enough to take account of all the factors that are relevant to the design of interactive systems.

In Human Reliability Assessment, where error likelihood quantification is a focus, further information about context is encoded in *performance shaping factors*. These are additional influences that affect the probability of certain human errors occurring. One way of looking at these probabilities is as a means of assessing the significance of some of the mutations discussed above and hence further limitation of the space of possible errors that must be discussed. Examples of performance shaping factors include the type of task and other general contextual circumstances. There are a variety of task classifications: for example does the task require vigilance; does it require problem solving; does it involve control, see [18] for further details. The general circumstances in which the task is being performed include workload [32] and situation awareness. Workload has been measured taking account of other tasks that are being carried out at the same time as well as the number of perceptual and articulatory channels that are being used by the operator. In practice these measures of context are highly subjective and therefore any attempt at quantification may not be reliable.

5 A wider understanding of context

The approach illustrated above is of course limited. The question that should be answered is whether it is rich enough to be useful. Typically the tasks that the designer anticipates do not coincide with the reality of actual work practices. Work systems involve multiple participants each with different viewpoints and goals. The errors that are introduced by humans when they use the system has been the focus for the discussion of this paper, for example mistaking information on the display in high workload situations, but human errors might also be latent in the organisational context. Here the designers, the specifiers, the maintenance crews who are also prone to human error might have a major influence on the safety of the system. The techniques that have been described above still do not capture all the important errors that can arise. By way of conclusion two examples of problems are considered that would not have been covered by the techniques described above. The first is described in more detail in [34].

In the first case a Lockheed Tristar suffered loss of oil pressure in all three engines; two engines failed and the third threatened to fail. On investigation it was discovered that, habitually, the specified maintenance procedure was not followed correctly. The foreman, who was expected to have a monitoring role, had taken responsibility for fitting O-rings during reassembly of the engines. During the maintenance preceding the incident in question the foreman was sick and though the check procedure was carried out, largely as documented, the subtask that had been usurped by the foreman was ignored.

The problem that arose was latent and required an analysis of the maintenance tasks that were carried out. Agency and responsibility are not a feature of the task descriptions that have been described in this paper. In fact these notions do not feature substantially in any task analysis notations. Given this limitation, an assessment of the risk of the conjunction of three omissions of a “fit O-ring action” would have been deemed to be acceptable, without any recognition that the same person would be doing all three actions in the context of one subgoal. It has already been argued that task descriptions provide a benchmark against which actual practice can be measured. What happens in practice is that the performance of the task may be modified. Even though checks were part of this task to ensure that the engines were reassembled correctly, these checks became less effective as the task mutated. This example shows that it is not only necessary that agency and responsibility be understood in task representation but also the mechanisms by which tasks are “optimised” over time.

These task mutations, or *work arounds* occur quite frequently in safety critical systems and not just “behind the scenes”. Another example has been found in aircraft where flap/slat retraction protection is provided. Here the operation of slat retraction is inhibited in circumstances where there is a high angle of attack or low air speed. Pilots have been known to work around their normal task by retracting slats immediately on take-off and assuming that the automatic system will actually retract them when angle of attack and speed are appropriate. Here the automation was designed as a fail check mechanism. Any assessment of the reliability of the system would have assumed a task performance in which early retraction was appropriately unlikely. This assumed redundancy, perhaps a pivotal component of a safety assessment, would not exist in practice.

6 Conclusions

A key emphasis of current research in Human Computer Interaction can be summed up in Kuutti’s words ([23]): “Before an action is performed in the real world it is typically planned in the consciousness using a model. The better the model the more successful the action. ... Models and plans are not rigid and accurate descriptions of the execution steps but always incomplete and tentative resources.”

The state of Human Computer Interaction is that theories are being sought to support an appropriate understanding of these models and plans as a significant influence in the design process. In the preface to Nardi’s edition [26], she states “There is a fundamental need of a theory of practice in human-computer interaction studies”. This paper has taken systems design and specification as a starting point for such an approach. We agree with writers such as Suchman, Nardi and Hutchins who argue that practical systems design techniques do not take enough account of the broader work system. We agree that simplistic views of procedures for using systems are dangerous. The main content of our argument is that designers require an approach which enables analysis and reasoning about the artefact taking an appropriate view of the requirements imposed by the surrounding work system.

The development described in this paper is continuing. Recent research, aimed at remedying deficiencies in our techniques, addresses issues of agency and responsibility in tasks, and the problem of capturing the context using more expressive methods than tasks. Interaction framework [16, 3] is being developed with the aim of providing a neutral framework in which agency and goal achievement can be analysed. A model of resources [35, 9] is being developed which captures, in the notion of action, not only state of the device, but also plans and other resources required in the broader context of understanding how the action is performed. The problem of course is to produce a practical and cost effective method for designing interactive systems. We are actively exploring the practicality and scaleability of this range of methods with industry driven case studies.

Acknowledgements

This work has been part funded by British Aerospace plc within the Dependable Computing Systems Centre, by Esprit III project Amodeus and two EPSRC grants GR/J08836 and GR/J07686.

References

[1] Abowd, G. and Dix, A. Giving undo attention. Interacting with Computers, 4(3):317–342, 1992.

[2] Bicarregui, J., Fitzgerald, J., Lindsay, P., Moore, R., and Ritchie, B. Proof in VDM: A Practitioner’s Guide. FACIT Series. Springer-Verlag, 1994.

[3] Blandford, A., Harrison, M., and Barnard, P. Using interaction framework to guide the design of interactive systems. International Journal of Human-Computer Studies, 43:101–130, 1995.

[4] Dearden, A. and Harrison, M. Formalising human error resistance and human error tolerance. In Proceedings of the Fifth International Conference on Human-Machine Interaction and Artificial Intelligence in Aerospace. EURISCO, 1995.

[5] Dearden, A. and Harrison, M. Impact as a human factor in interactive system design. In Redmill and Anderson (Eds), Safety Critical Systems: The Convergence of High Tech and Human Factors, pages 184–199. Springer, 1996.

[6] Dix, A. Formal Methods for Interactive Systems. Academic Press, 1991.

[7] Duke, D. and Harrison, M. A preliminary FSM analysis of the CERD. Technical Report SYSMOD IR 3, MRC-APU, Cambridge — Amodeus 2 Working Papers, 1994.

[8] Duke, D. and Harrison, M. A theory of presentations. In FME’94: Industrial Benefit of Formal Methods, volume 873 of Lecture Notes in Computer Science, pages 271–290. Springer-Verlag, 1994.

[9] Fields, B., Wright, P., and Harrison, M. Designing human-system interaction using the resource model. In APCHI’96: First Asia Pacific Conference in Human Computer Interaction, pages 181–191, Singapore, June 1996. Information Technology Institute.

[10] Fields, R., Wright, P., and Harrison, M. A task centered approach to analysing human error tolerance requirements. In Proceedings of IEEE Symposium RE’95, pages 18–26, 1995.

[11] Fields, R., Wright, P., and Harrison, M. A method for user interface development in safety-critical applications. Technical report, University of York, 1996.

[12] Grudin, J. The case against user interface consistency. Communications of the ACM, 4(3):245–264, 1989.

[13] Hall, A. Using Formal Methods to develop an ATC information system. IEEE Software, 33(2):66–76, March 1996.

[14] Harrison, M. D., Roast, C. R., and Wright, P. C. Complementary methods for the iterative design of interactive systems. In Salvendy, G. and Smith, M. (Eds), Designing and Using Human-Computer Interfaces and Knowledge Based Systems, pages 651–658. Elsevier Scientific, 1989.

[15] Harrison, M. Engineering Human Error Tolerant Software. In Nichols (ed), Proceedings of the Z User Conference, pages 191–204. Springer-Verlag, 1992.

[16] Harrison, M., Blandford, A., and Barnard, P. The requirements engineering of user freedom. In Paternò, F. (ed), Proc Eurographics Workshop on Design Specification and Verification of Interactive Systems, Italy, pages 181–194. Springer-Verlag, 1995.

[17] Hoare, C. A. R. Communicating Sequential Processes. Prentice Hall International, 1985.

[18] Hollnagel, E. Reliability of cognition: foundations of human reliability analysis. Academic Press, 1993.

[19] Hutchins, E. Cognition in the Wild. MIT Press, 1994.

[20] Jones, C. B. Systematic Software Development Using VDM (2nd ed.). Prentice-Hall, 1990.

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

[22] Kirwan, B. and Ainsworth, L. A Guide to Task Analysis. Taylor and Francis, 1992.

[23] Kuutti, K. Activity theory as a potential framework for human-computer interaction research. In Nardi, B. (ed), Context and Consciousness: Activity Theory and Human Computer Interaction, pages 17–44. MIT Press, 1996.

[24] MacKenzie, D. Computer-related accidental death: an empirical exploration. Science and Public Policy, 21(4):233–248, Aug 1994.

[25] Milner, R. Communication and Concurrency. Prentice Hall International, 1989.

[26] Nardi, B. (ed). Context and cognition: a theory of human computer interaction. MIT Press, 1996.

[27] Norman, D. A. The Psychology of Everyday Things. Basic Books, 1988.

[28] Roast, C. Executing Models in Human Computer Interaction. PhD thesis, Dept. Computer Science, University of York, 1993.

[29] Saiedian, H. An invitation to formal methods. IEEE Computer, pages 16–17, April 1996.

[30] Salminen, S. and Tallberg, T. Human errors in fatal and serious occupational accidents in Finland. Ergonomics, 39(7):980–988, 1996.

[31] Suchman, L. Making work visible. Communications of the ACM, 38(9):56–64, 1995.

[32] Wickens, C. Engineering Psychology and Human Performance. Charles E. Merrill Publishing Company, 1984.

[33] Winograd, T. and Flores, F. Understanding computers and cognition: a new foundation for design. Addison-Wesley, 1987.

[34] Woods, D. D., Johannesen, L. J., Cook, R. I., and Sarter, N. B. Behind human error: Cognitive systems, computers, and hindsight. Technical Report SOAR 94-01, Crew Systems Ergonomics Information and Analysis Center (CSERIAC), Wright- Patterson Airforce Base, Ohio., 1994.

[35] Wright, P., Fields, B., and Harrison, M. Distributed information resources: An new approach to interaction modelling. In Proceedings of ECCE8: European Conference on Cognitive Ergonomics. EACE, 1996. To appear.