# From a Formal User Model to Design Rules

Paul Curzon[1] and Ann Blandford[2]

[1]Middlesex University, Interaction Design Centre, Bramley Road, London N14 4YZ
[2]University College London Interaction Centre, 26 Bedford Way, London WC1H 0AP
p.curzon@mdx.ac.uk, a.blandford@ucl.ac.uk

**Abstract.** Design rules sometimes seem to contradict. We examine how a formal description of user behaviour can help explain the context when such rules are, or are not, applicable. We describe how they can be justified from a formally specified generic user model. This model was developed by formalising cognitively plausible behaviour, based on results from cognitive psychology. We examine how various classes of erroneous actions emerge from the underlying model. Our lightweight semi-formal reasoning from the user model makes predictions that could be used as the basis for further usability studies. Although the user model is very simple, a range of error patterns and design principles emerge.

## 1 Introduction

With the increasing ubiquity of interactive computer systems, usability becomes increasingly important. Minor usability problems can scale to having major economic and social consequences. Usability of interactive designs has many aspects. In this paper, we focus on design principles that reduce the potential for "user error" occurring. We examine how, from the behaviour specified by a simple formal model of cognition, various potential erroneous actions emerge with poorly designed systems. Furthermore, we derive well-known design rules from the model. We use the fact that the rules are grounded in a formal model of cognition to explore the scope of application of the rules. Formal specification allows precision about the meaning of that which is formalised. Providing such precision to ensure different people have the same understanding of a concept has been suggested as the major benefit of formal models in interaction design [1]. One approach would be to formalise the design rules themselves (see [1, 14]). Here, we semi-formally derive design rules from a formal model rather than just asserting them. In principle, formal derivations could also be done. By "semi-formal" we mean that we use informal high-level argument (about a fully formal model), as opposed to the explicit application of inference rules of the underlying logic.

We first define simple **principles of cognition**. These are principles that generalise the way humans act in terms of the mental attributes of knowledge, tasks and goals. The principles considered do not cover the full range of human cognition. Rather they focus on particular aspects of cognitive behaviour. They are each backed up by evidence from HCI and psychology studies. Those presented are not intended to be complete but to demonstrate the approach.

We have developed a **formal model** of these principles written in higher-order logic. This description is a generic formal user model. By "generic" we mean that it can be targeted to different tasks and interactive systems. The underlying principles of cognition are formalised once in the model, rather than having to be re-formalised for each new task or system of interest. Whilst higher-order logic is not essential for this, its use makes the formal specifications simpler and more natural than the use of a first-order logic would. Here we use it to make precise the general principles considered, to allow us to then reason about their consequences with respect to user error. Combining the principles of cognition into a single model rather than formalising them separately allows reasoning about their interaction, and about how multiple minor errors might interact.

The principles, and more formally the user model, specify **cognitively plausible behaviour** (see [5]). That is, they specify possible traces of user actions that can be justified in terms of the specific principles. Of course users might also act outside this behaviour, about which situations the model says nothing. Its predictive power is bounded by the situations where people act according to the principles specified. That does not preclude useful results from being obtained, provided their scope is remembered. The model allows us to investigate what happens if a person does act in such plausible ways. The behaviour defined is neither "correct" nor "incorrect". It could be either depending on the environment and task in question. It is, rather, "likely" behaviour.

We show how cognitively plausible behaviour can in specific circumstances be considered as resulting in **erroneous actions**. We discuss the circumstances in which such erroneous actions can result, reasoning from the formal model. In particular, we relate them to Hollnagel's error phenotypes [11]. In doing so, we identify cognitively plausible ways in which the erroneous actions could arise. We show that a wide variety are covered from even a minimal formal definition of cognitively plausible behaviour, demonstrating the generality of the approach.

Finally, we semi-formally derive **design rules** from the formal model that, if followed, ensure that the erroneous actions identified will not be made for the specific cognitive reasons embodied by the principles of cognition. Even though the user model is capable of making the errors, the rules ensure that the environments in which they would emerge do not occur. Other errors are, of course, still possible. The design rules are well known and our contribution is not the rules themselves, but rather the demonstration that they can be justified from a formalisation of a small set of principles. Because the design rules are derived, we can be precise about their scope of applicability, for example unpacking the situations where different design rules appear at first sight to be contradictory.

We use railway ticket vending machines to illustrate the points. They are ubiquitous and are increasingly replacing manned ticket offices. However, existing designs continue to exhibit design problems that encourage user error [16]. Previous work explored how the user model considered here could be used to analyse interactive system designs by treating it as a component of that system with a fixed design [7, 8], proving that a specific task will always be completed eventually. This was done using an interactive proof system, HOL [10]. Here we use the user model as the basis of reasoning about interactive systems in general. The process of doing so also acts, in part, to validate the model for formal verification. Our approach is similar to that of [2] in that we are working from a (albeit different and more formal) model of user behaviour to high level guidance. There the emphasis is on a semi-formal basis underpinning the craft skill in spotting when a design has usability problems. In contrast, we are concerned with guidance for a designer rather than for a usability analyst.

## 2 Formalising Cognitively Plausible Behaviour

Our user model was developed by formally modelling principles of cognitively plausible behaviour. We do not model erroneous actions explicitly (as is done for example in [9]). Instead, they emerge from an abstract description of cognitively plausible behaviour. The behaviour described could correspond to correct or incorrect actions being taken depending on the circumstances. The principles considered are: non-determinism; goal-based termination; task-based termination; reactive behaviour; communication goals; mental triggers; no-option-based termination; and relevance. This list is not intended to be exhaustive, but to cover a variety of classes of cognitive principles, based on the motor system, simple knowledge based cognition, goal-based cognition, *etc.* Also, some of the principles are formalised in a simple way, our intention at this stage being to test the approach, rather than modelling the full richness of the principles. In future work we will increase the richness of the descriptions. In subsequent sections we refer to cognitively plausible behaviour when strictly meaning the subset of cognitively plausible behaviour embedded in the current version of our model.

By modelling the principles we are giving a *knowledge level* description in the terms of Newell [12]. We do not attempt to model underlying neural architecture nor the higher level cognitive architecture such as working memory units. Instead our model is that of an abstract specification, intended for ease of reasoning. The focus of the description is in terms of internal goals and knowledge of a user. This contrasts with a description of a user's actions as, say, a finite state machine that makes no mention of such cognitive attributes.

The user model is based on a series of non-deterministic temporally guarded action rules. Each describes an action that a user *could* rationally make. The rules are grouped corresponding to the user performing actions for specific cognitively related reasons. Each such group then has a single generic description. Each rule has a guard-action form. They state that if the guard holds at some point then the `NEXT` action taken by the user is that given. By *next* in this context we mean the first action of interest taken by the user after the current point in time. The rules each have the form: `guard t AND NEXT actions action t`, stating that a `guard` is true at time `t` and the `NEXT` action performed from the list of actions relevant to the interaction (given by the list `actions`) is `action`. The action is identified by its position in the list of all actions. Here we give an overview; the formalisation is given in more detail in [8].

**Non-determinism:** In any situation, any one of several behaviours that are plausible might be taken. The separate behaviours are specified as rules. Each rule is formalised in the user model non-deterministically. That is, it is one of a series of options, any of which could be taken. The model does not assert that a rule will be followed, just that it may be followed. Below, we present the formalisation of several such rules. They form the core of the user model. By combining them, the model asserts that the behaviour of any rule whose guards are true at a point in time is cognitively plausible at that time. It cannot be assumed that any specific rule will be the one that the person will follow.

**Goal-based termination behaviour:** Cognitive psychology studies have shown that users intermittently, but persistently, terminate interactions as soon as their goal has been achieved [6]. It is formalised as a guarded rule as described above. We must supply a relation to the user model that indicates over time whether the goal is achieved or not. This is referred to as a special signal, `goalachieved`, in the formal definitions. We also use a special signal, `finished`, to indicate whether the user considers the interaction to be over. With a ticket machine this may correspond to the person walking away, starting a new interaction (perhaps by hitting a reset button), *etc.* Both `goalachieved` and `finished` are signals that, given a time, return a boolean value indicating whether the goal is achieved or the interaction terminated (respectively) at that time. If the goal is achieved at a time then the user model terminates the interaction next: `goalachieved t AND NEXT actions finished t`. Note that `goalachieved` is a higher-order function and can as such represent an arbitrarily complex condition. It might, for example, be that the user has a particular object, that the count of some series of objects is greater than some number or a combination of such atomic conditions. In specifying the user model we just state that it is a boolean function whose value may vary over time. This makes use of the higher-order nature of the specification language.

**Task-based termination behaviour:** For the purposes of analysis, the model specifies that a user will terminate an interaction when their whole task is achieved. In achieving a goal, subsidiary tasks are often generated. For the user to complete the task associated with their goal they must also complete all subsidiary tasks. Examples of such tasks with respect to a ticket machine include taking back a credit card or taking change [6]. One way to specify these tasks would be to explicitly describe each such task. Instead we use the more general concept of an interaction invariant [8]. The underlying reason why these tasks must be performed is that in interacting with the system some part of the state must be temporarily perturbed in order to achieve the desired task. Before the interaction is completed such perturbations must be undone. For example, to pay at a ticket machine using a credit card requires the card being inserted and later returned. A condition on the state that holds at the start of the interaction – that the user has the card – must be restored by the end. We specify the need to perform these completion tasks indirectly by supplying the interaction invariant as a higher-order argument to the user model. The interaction invariant is an invariant in a similar sense to a loop invariant in program verification. It is an invariant at the level of abstraction of whole interactions. Full task completion involves not only completing the user's goal, but also restoring the invariant by completing all the subsidiary tasks generated in the process. For a ticket machine the invariant might specify that the value of a person's possessions at the end is at least as high as it was at the start of the interaction.

We assume that on completing the task in this sense of goal achieved and invariant restored, the interaction will be considered terminated by the user, irrespective of any other possible actions apart from actions already mentally triggered (discussed below). This is modelled using an if construct rather than disjunction to give it priority. If both the goal has been achieved and the invariant restored then the user will terminate the interaction, irrespective of what other non-deterministic rules may potentially be active. Otherwise one of the non-deterministic rules will be fired.

```
IF (invariant t) AND (goalachieved t) THEN NEXT actions finished t
                                      ELSE non-deterministic rules
```

**Reactive behaviour:** A user may react to a stimulus or message from a device, doing the action suggested by the stimulus. For example, if a flashing light comes on next to the coin slot of a ticket vending machine, a user might, if the light is noticed, react by inserting coins if it appears to help the user achieve their goal. Reactive behaviour is specified as a general class of behaviour: in a given interaction there may be many different stimuli to react to. Rather than specify this class of behaviour for each, we define the behaviour generically. `REACT` gives the rule defining what it means to react to a given stimulus.

```
REACT as stimulus action t = stimulus t AND NEXT as action t
```

If at time `t`, the specified `stimulus` is active, the NEXT action taken by the user out of the possible actions, `actions`, at an unspecified later time, may be `action`. As there may be a range of signals designed to be reactive, the user model is supplied with a list of stimulus-action pairs: [(s1, a1); . . . (sn, an)]. A list recursive relation, given a list of such pairs, extracts the components and asserts the above rule about them. They are combined using disjunction in the recursive definition, so are non-deterministic choices, and this definition is combined with the other non-deterministic rules. `Grd` and `Act` extract a pair's components. "`s :: st`" refers to the list with first element `s` and remainder of list `st`.

```
(REACTS as [] t = FALSE) AND
(REACTS as (s :: st) t =
   ((REACTS as st t) OR (REACT as (Grd s) (Act s) t)))
```

**Communication goal behaviour:** A user enters an interaction with knowledge of task dependent sub-goals that must be discharged. Given the opportunity, they may attempt to discharge any such *communication goals* [3]. The precise nature of the action associated with the communication goal may not be known in advance. A communication goal specification is a task level partial plan. It is a pre-determined plan that has arisen from knowledge of the task in hand independent of the environment in which that task will be accomplished. It is not a fully specified plan, in that no order of the corresponding actions may be specified. In the sense of [3] a communication goal is purely about information communication. Here we use the idea more generally to include other actions that are known to be necessary to complete a task. For example, when purchasing a ticket, in some way the destination and ticket type must be specified as well as payment made. The way that these must be done and their order may not be known in advance. However, a person enters an interaction with the aim of purchasing a ticket primed for these communication goals to be addressed. If the person sees an apparent opportunity to discharge a communication goal they may do so. Once they have done so they will not expect to need to do so again. No fixed order is assumed over how communication goals will be discharged if their discharge is apparently possible. For example, if a "return ticket" button is visible then the person may press that first if that is what they see first. If a button with their destination is visible then they may press it first. Communication goals are a reason why people do not just follow instructions.

Communication goals are modelled as guard-action pairs as for reactive signals. The guard describes the situation under which the discharge of the communication goal appears possible. It will include a label signal indicating that the input exists and that it corresponds to the desired action. In the current version of the model, the use of a special label signal is not built into the generic model but is included as part of the guard by convention. As for reactive behaviour, a list of (guard, action) pairs is supplied to correspond to each communication goal. A similar recursive definition to `REACTIVE` above is defined and included as a disjunct with the non-deterministic rules. This determines when a communication goal may be discharged. However, unlike the reactive signal list that does not change through an interaction, communication goals are discharged. This corresponds to them disappearing from the user's mental list of intentions. We model this by removing them from the communication goal list when done. A daemon, separate from the non-deterministic rules, does this. It monitors the actions taken by the user on each cycle, removing any from the list used for the subsequent cycle. The action removed may be taken for some reason other than it being a communication goal, such as due to reactive behaviour. All that matters to the daemon is that it is taken. The communication goal list that a user enters the interaction with initially is provided as an argument to the user model.

**Mental triggers:** A user commits to taking an action in a way that cannot be revoked after a certain point. Once a signal has been sent from the brain to the motor system to take an action, the signal cannot be stopped even if the person becomes aware that it is wrong before the action is taken. Rather than associate an external stimulus directly with an external action using the disjunctive rules, we associate them with mental "actions" that trigger the process of taking the actual action. Thus the actions in each of the rules described so far will not be externally visible actions, but internal mental actions. For example, on deciding to press a button labelled with the destination “Liverpool”, at the point when the decision is made the mental trigger action takes place and after a very short delay, the actual action takes place. A further category of trigger rules is then introduced that links the mental decision to the actual action. If one of the mental actions is taken on a cycle then the next action will be the externally visible action it triggers. There is always at least a one-cycle delay between the trigger and external action. A recursive function combines a list of triggers into a series of choices as with the reactive rules. The user model must be supplied with a guard-action pair list linking mental triggers with external actions. As with task-based termination, mental triggers are given a higher priority than the non-deterministic rules. If a trigger is fired then it will be the next action taken. Only if no fired trigger is outstanding do the other rules come into play, including task-based termination.

**No-option-based termination behaviour:** A user may terminate an interaction when there is no apparent action they can take that would help complete the task. For example, if on a touch screen ticket machine, the user wishes to buy a weekly season ticket, but the options presented include nothing about season tickets, then the person might give up, assuming their goal is not achievable. The model includes a final default non-deterministic rule that models this case. The guard to this rule is constructed automatically in the model from the information supplied to create the other rules. In practice, in this situation, people could behave in a range of ways including pressing buttons at random. Our model treats a situation where no “rational” action is available as resulting in the interaction terminating – even if a possible action may become possible in the future. Note that a possible action that a person could take is to wait. However, they will only do so given some reason – that is, it must be an action in an explicit reactive rule. For example, a ticket machine might display a message “Please Wait”. If they see it, the person reacts by waiting.

**Relevance:** A user will only take an action if there is something to suggest it corresponds to the desired effect. We do not currently model this explicitly: however, it is implicit in most of the rules. For example, communication goals and the termination rules are by definition only fired when relevant. In particular, the “label” signals referred to above are intended to address aspects of relevance. A button for the destination “Liverpool” is modelled by one signal representing whether the button is visible/relevant at a given time and a second about whether the button is pressed at each time instance.

**Putting it together:** The core rules are combined with other house keeping rules (most notably, the communication goal filtering daemon) and a model of possessions that specifies, for example, that a user ceases to have a possession if it is given up. We omit the details here due to space constraints. A further clause added to the model is the initial conditions – notably the initial communication goal list. These are all combined using conjunction into a single relation `USER` that models the full user model. It takes as arguments the various pieces of information such as the goal, interaction invariant, list of actions, *etc.* referred to in the description above.

## 3 The Erroneous Actions that Emerge

Erroneous actions are the proximate cause of failure attributed to human error in the sense that it was a particular action (or inaction) that immediately caused the problem: users pressing a button at the wrong time, for example. However, to understand the problem, and so ensure it does not happen again, approaches that consider the proximate causes alone are insufficient. It is important to consider why the person took that action. The ultimate causes can have many sources. Here we consider situations where the ultimate causes of an error are that limitations of human cognition have not been addressed in the design. An example might be that the person pressed the button at that moment because their knowledge of the task suggested it sensible. Hollnagel [11] distinguishes between human error `phenotypes` (classes of erroneous actions) and `genotypes` (the underlying psychological cause). He identifies a range of simple phenotypes: repetition of an action, reversing the order of actions, omission of actions, late actions, early actions, replacement of one action by another, insertion of an additional action from elsewhere in the task, and intrusion of an additional action unrelated to the task. These are single deviations from required behaviour.

In practical designs it is generally infeasible to make erroneous actions impossible. Fields [9] uses model-checking to identify errors by introducing the above problems explicitly into task specifications. A problem with this approach is that it gives many false negatives: few tasks are possible if such errors are arbitrarily made. The verifier must determine which are real problems. A definition of what is cognitively plausible is one way to make this judgement. A more appropriate aim is therefore to ensure that *cognitively plausible* erroneous actions are not made. To ensure this, it is necessary to consider the genotypes of the possible erroneous actions. We examine how our simple user model can exhibit behaviour corresponding to these errors. We thus show, based on reasoning about the formal model, that, from the minimal principles we started with, a wide range of classes of erroneous actions in the form of phenotypes occur.

We now look at each simple phenotype and at the situations where they are cognitively plausible. We do not claim to model *all* cognitively plausible phenotypical actions. There are other ways each could occur for reasons we do not consider. However, not all errors that result from the model were explicitly considered when the principles were defined. The scope of the model in terms of erroneous actions is wider than those it was originally expected to encompass.

**Repetition of actions:** The first class of erroneous action is to repeat an action already performed. There are situations where this is cognitively plausible according to our user model. The current user model will repeat actions if guided to do so by the device in a reactive manner. If the guards of an action remain true then the user model may follow those instructions a second time since there is nothing in the model to prevent this. If the guidance is erroneous then the user model will make an erroneous action. Occasions where an interactive device asks erroneously for an action that has already been performed are perhaps rare (and it might be argued that in this situation the action was correct but the device incorrect). However, one way it could occur is due to a lack of feedback to indicate the action was performed successfully. The current user model would do this if reactive signals guided the action and continued to do so after the action had been completed. In particular, with a ticket machine, if a light next to a coin slot continued to flash for a period after the correct money had been inserted a person might assume they had not inserted enough and start to insert more. An action originally performed as a communication goal could be repeated if a reactive prompt to do so later appeared (though not the other way round since once performed reactively the action is removed as a communication goal). For example, if a person pressed the button for "Liverpool" and was later presented with a screen asking them to select a destination they might do so again.

**Reversing the order of actions:** A second class of error is to reverse the order of two actions. This pattern of behaviour can arise from our model as a result of the way communication goals are modelled. In particular, communication goals can be discharged by the user model in any order. Therefore, if an interactive system requires a particular sequence, then the order may be erroneously reversed by the user model if the actions correspond to communication goals. A person might insert money and then press the destination button when a particular ticket machine requires the money to be inserted second. This does not apply to non-communication goal actions, however. For example, two actions that are device dependent (pressing a confirmation button and one to release change, for example) will not be reversed by the user model.

**Omission of actions:** The user model may omit actions at the end of a sequence. In particular, it may terminate the interaction at any point once the goal has been achieved. For example, once the person is holding the ticket they intended to buy, they may walk away from the machine, leaving their change, credit card or even return portion of their ticket. Whatever other rules are active, once the goal is achieved, the completion rule is active, so could be fired. The user model may also omit trailing actions if there is no apparent action possible. If at any time instance the guard of no other rule is active, then the guard of the termination rule becomes active and so the user model terminates. There must always be some action possible. This could be to pause but only if given reactive guidance to do so. For example, if there is a period when the ticket machine prints the ticket, where the person must do nothing, then with no feedback they may abort. In this respect the user model does not quite reflect the way people behave. If there is no action possible the user model is guaranteed to terminate, whereas in reality a person might pause before giving up. However, if the concern is user error, this is not critical as either way termination is possible so task completion is not guaranteed. If the user model took an action early due to it corresponding to a communication goal (e.g. selecting a destination first instead of ticket type) then the model would assume that the action had had the desired effect. The action (selecting a destination) would be removed from the communication goal list: the model "believes" it has been performed. It then would not be done at the appropriate point in the interaction i.e., a second (omission) error would occur. In this situation the correct action would be a *repetition* of the earlier action – repetition is not an error in this situation.

**Late actions:** The user model does not put any time bounds on actions. All rules simply assert that once an action is selected then it will eventually occur. If any action must be done in a time critical manner, then the user model will be capable of failing to do so. In practice this is too restrictive – it means the current user model will always be able to fail with a device that resets after some time interval, for example, as would be normal for a ticket machine. Where such time criticality is inherent in a design, extra assumptions that deadlines are met would need to be added explicitly.

**Early actions:** If there are periods when an action can apparently be performed, but if performed is ignored by the computer system, then in some circumstances the user model would take the next action early. In particular, if the user has outstanding communication goals then the corresponding actions may be taken early. This will potentially occur even if the device gives explicit guidance that the user must wait. This corresponds to the situation where a person does not notice the guidance but takes the action because they know they have to and have seen the opportunity. Similarly, if the device is presenting an apparent opportunity for reactive behaviour before it is ready to accept that action then the user model could react to it.

**Replacement of one action by another:** Replacement can occur due to communication goals if the device requires a specific action to be taken but its interface suggests that a communication goal can be discharged. For example, if the coin slot is visible but a destination selection required first, the person may insert money as discussed earlier. The user model may make the communication goal action rather than the required one, even if instructions are being displayed. Similarly, if reactive signals give incorrect guidance that suggests an action should be taken then that guidance may be followed. It can also occur due to trigger rules and environmental changes. In particular, if a change of state in the computer system can occur, not in response to a user action, then if the user model has already committed to some action (such as pressing a button), but its effect changes between the commitment being made and the action actually being taken, then the wrong effect will occur. This can lead to a person doing something they know is wrong. The change could occur due to a machine time-out or an environmental change (e.g. the time changing to off-peak travel).

**Insertion of actions from elsewhere in the task:** Insertion of an action can occur with communication goals. They can be attempted by the user model at any point in the interaction where the opportunity to discharge them apparently presents itself. With reactive tasks, it will occur only if the device gives a reactive signal to suggest it can be done when it cannot.

**Intrusion of actions unrelated to the task:** Actions unrelated to the task can intrude with the user model as a result of reactive signals on the device. If a device supports multiple tasks and uses reactive signals that signal an action to be performed that is not part of the task, such an action may be taken.

In summary, the principles of cognition implemented in the model generate behaviours that account for Hollnagel's various phenotypes. Similarly, those same principles of cognition can be used to derive and reason about design principles.

## 4 Design Rules

We now examine some usability design rules and how they solve the problems identified. Ad-hoc lists of design rules can easily appear to be contradictory or only apply in certain situations. By basing them on cognitively plausible principles, we can reason about their scope and make this scope more precise. For example, should systems always be permissive [15], allowing any action to be taken, or only under certain circumstances? Permissiveness appears to contradict forcing functions [13] when only certain actions are made possible. By reasoning from cognitive principles we can untangle these surface contradictions.

**Completion actions:** The user model contains a rule to terminate if the goal is achieved. Whatever other rules are active, this one could be activated due to the non-deterministic nature of the rules. The user model can therefore terminate the moment its goal is achieved. Furthermore, no output from the device can prevent this as it would just result in additional rules being active which cannot preclude some other action being taken. For the user model to guarantee to not terminate early for this reason it must only be possible for a user to terminate once the task is completed. Thus for our user model the task must be completed no later than the goal. Any design that requires the user model to perform extra completion tasks must ensure they are done before the goal is achieved. The rule will then only be active precisely when the task termination rule will be active, so that termination does not occur before the task rule is achieved. In practice (e.g. when termination involves logging out from a system) it may not always be possible to satisfy this design rule; in such situations, another means of restoring the invariant needs to be found. An attempted verification of a design that did not follow this design rule would fail because there would be a path where the goal was achieved and so termination would occur on that path, when the task was not achieved. In particular, as noted above, providing extra information is not sufficient. For a ticket machine, taking the ticket must be the last action of the user. They must by then have taken change or hold their credit card, or these must be returned in the same place and at the same time as the ticket. Multiple ticket parts (e.g. the return ticket) must also be dispensed together.

**Provide information about what to do:** Actions that are not communication goals can only be triggered in the model if they are a response to reactive signals – information indicating that the given rule is the next to be performed to achieve the given task. Therefore, if an action must be performed that does not correspond to a communication goal then information in the form of clear reactive guidance needs to be provided to tell the user to take the action. In the case of a ticket machine, if a button must be pressed to confirm the ticket selected is the one required, then instructions to do this must be provided. For communication goal actions, reactive information is not needed, though information linking the communication goal to the specific action is needed: something (such as the presence of a visible coin slot for inserting money) must make it clear that the communication goal can be discharged.

**Providing information is not enough:** The above design rule concerned always providing information. This one is that that is not good enough – so might appear to be contradictory. However, it depends on the situation. A simple design rule might be to clearly indicate the order that actions should be taken. This approach is often used where, for example, a panel gives instructions or lights flash to indicate the next button to press. However, the user model is non-deterministic. There may be several rules active and therefore several possible actions that could be taken. Reactive signals are not modelled as having higher priority than any other signal. Other possible actions are, for example, to terminate the interaction (if the goal is achieved), or discharge a communication goal. If the guards of such rules are active then they are possible actions. Making other signals true cannot make such a guard false; it can only make false guards true, so increasing the range of possible actions. Therefore, just providing flashing lights or beeps or other reactive signals is not enough to ensure correct operation. An attempted verification of such a design would fail because it would not be possible to prove that the correct action was taken. Some other action would be possible which could ultimately lead to the user aborting the interaction. If any possible path leads to abortion before the goal is achieved then the correctness statement will be unprovable as it states that the goal is achieved on all paths. Is providing information ever enough? According to the model – yes. It is sufficient if the user has nothing else to do and the action clearly takes them towards their goal. Thus (for our principles) if all communication goals are discharged (the ticket has been specified and money inserted) and the goal is not achieved (no ticket is held) then providing information is useful and necessary.

**Forcing functions:** The fact that the user model is capable of taking several different options and that giving reactive signals and messages is not enough means that some other way is needed to ensure the options are narrowed down to only the correct ones. As Norman [13] suggests, in good design, only correct actions for the range of tasks supported at a point should be possible. This suggests the use of forcing functions. Somehow the design must ensure that the only cognitively plausible actions are correct ones. This does not mean there must only be one button to press at any time, but only one button that can possibly be of use. Within the limits of the model, this means that if communication goals are not yet discharged, and should not yet be discharged, then there should be no apparent opportunity to discharge them. For example, a soft screen might be used so that the only buttons pressable correspond to ones that can now correctly be pressed. If money cannot be inserted then the coin slot should be closed. Similarly, the solution to post-completion errors is to not allow the goal to be achieved until the task is completed – forcing the user to complete other completion tasks first (where possible), as discussed above.

**Permissiveness:** Forcing functions follow the design principle that the options available to the user should be reduced. An alternative way of solving the same problem is to do the opposite and make the design permissive [15]: that is, it does not force a particular ordering of events. In this case, the design should be such that each of the actions that can be taken by the user model are accepted by the design and lead to the task being achieved. With our user model, permissiveness cannot be used universally, however. For example, it is not sufficient with completion tasks to allow them to be done in any order. As we have seen, if the goal is achieved before the task is completed then the user model leaves open the possibility of termination. There is no way the design can recover – once the user model terminates it does not re-start the task. Therefore, in this situation, being permissive does not work. The ticket must be released last. That action corresponds to the goal so cannot be permissive. At times in an interaction when communication goals are outstanding, the user model could discharge them if the opportunity is present. Thus permissiveness is a useful design rule to apply to communication goals. In particular, permissiveness should be applied if forcing functions are not used when communication goals are active. A communication goal that appears dischargable should be dischargable. For example, a ticket machine could allow destination and ticket type to be chosen in any order.

**Visibility:** The user model provides for both reactive behaviour and directly goal-based behaviour. All user model actions are guarded by a signal indicating the presence of information suggesting it is an appropriate action. If a control is not labelled then the user model will not take the action. Thus all controls must be labelled if the user model is to use them. This does not mean that labels must be written. The form of a control may be considered sufficient to warrant the signal being asserted. For example, a coin slot advertises by its form that it is for the insertion of coins. This would need to be decided by a usability expert using complementary techniques. Also, it only needs to be visible at the point where the user model must take the action. Thus visibility need not be universal.

**Give immediate feedback:** If there is no possible action apparent to the model then it will abort. If a user must wait while a ticket is printed, then feedback to wait should appear immediately with nothing else apparently possible (e.g. no other buttons visible). One possible reactive action can always be to pause provided it is guarded by the existence of a "please wait" message.

**Do not change the interface under the user's feet:** The existence of trigger behaviour, where there is a delay between the user making a decision and acting on it, but after which they cannot stop themselves, leads to a design rule that the interface should not change except in response to user action. More specifically, a possible design rule is that no input to the computer system should change its meaning spontaneously. This is quite restrictive, however. Less restrictive design possibilities are available to overcome the problems. For example, most ticket machines have timeouts – if no action is made in some period then the machine resets to some initial state. The user model does not strictly support such behaviour at present. However, one possibility with the current limited user model, and as used by some cash points, is to ask the user if they want more time after some delay. However, this means the buttons change their meanings. What did mean "I want to go to Liverpool" suddenly means "I do not want more time", for example. Such problems can be overcome, provided the old buttons all mean "I want more time", and the one that means no more time previously was not linked to any action – or with a soft-button interface did not exist at all. Such a design would only work with the user model if reactive signals were being used, as if the action were taken as a result of a communication goal, then that communication goal would have been discharged. The user model would only take the action again if prompted reactively to.

**Where possible, determine the user's task early:** The user model can take reactive actions intended for other tasks. This can be overcome if multiple task devices determine the task to be performed at the first point of divergence between the tasks. For example, a ticket machine that can also be used as a cash point may have a common initial sequence inserting a credit card. However, once the tasks diverge, the next device action should be to determine the task the user is engaged in, in a way that makes no other actions (specifically communication goals for any of the tasks) apparently possible. From then on actions from other tasks will not need to intrude in the design. This is important since a communication goal can be discharged at any point where apparently possible. In complex situations this will be difficult to achieve.

## 5 Conclusions and Further Work

We have outlined a formal description of a very simple user model. The user model describes fallible behaviour. However, rather than explicitly describing *erroneous* behaviour, it is based on *cognitively plausible* behaviour. Despite this we show that a wide variety of erroneous actions can occur from the behaviour described in appropriate circumstances. We have considered how devices (software, hardware or even everyday objects) must be designed if a person acting as specified by the user model would be able to successfully use the device. We have shown how well-known design rules, if followed, would allow this to occur. Each of these rules removes potential sources of user error that would prevent the verification of a design against the user model using the techniques described in [7]. We thus provide a theoretically based set of design rules, built upon a formal model. This model has very precise semantics that are open to inspection. Of course our reasoning is about what the user model might do rather than about any real person. As such, the results should be treated with care. However, errors that the user model could make are cognitively plausible and so worth attention.

One of our aims was to demonstrate a lightweight use of formal methods. As such, we have started with a formal description of user behaviour and used it as the basis for *semi-formal* reasoning about what erroneous behaviours emerge, and the design principles that would prevent behaviours emerging. Such semi-formal reasoning could contain errors. We also intend to explore the formal, machine-checked derivation of the design principles. Using HOL (the proof system the user model is defined within), this would involve giving formal descriptions of design rules and proving that – under the assumptions of the user model – particular erroneous situations would not occur.

Our model is intended to demonstrate the principles of the approach and covers only a small subset of cognitively plausible behaviour. As we develop it, it will give a more accurate description of what is cognitively plausible. We intend to extend it in a variety of ways. As this is done, more erroneous behaviour will be possible. For example, habitual behaviour is currently not modelled.

Also many aspects of an interactive systems are parameters of the user model. However, generally a user actually determines this information by observation of the machine's interface. The model could be modified so that such information is an input in the model (i.e. collected as part of the interaction) rather than supplied by the verifier. We have essentially made predictions about the effects of following design rules. In broad scope these are well known and based on usability experiments. However, one of our arguments is that more detailed predictions can be made about the scope of the design rules, relating them back to concepts such as communication goals. The predictions resulting from the model could be used as the basis for designing further experiments to validate the model, or further refine it. We have also suggested there are tasks where it might be very difficult or even impossible to produce a design that satisfies all the underlying principles, so that some may need to be sacrificed in particular situations. We intend to explore this issue further.

## References

1. A.E. Blandford, P.J. Barnard and M.D. Harrison. Using Interaction Framework to guide the design of interactive systems. *International Journal of Human Computer Studies*, 43:101-130, Academic Press 1995.
2. A. Blandford, R. Butterworth and P. Curzon, Puma Footprints: Linking Theory and Craft Skill in Usability Evaluation. *Proc. Interact 2001*, pp577-584, IOS 2001.
3. A. Blandford and R. Young, The role of communication goals in interaction. In *Adjunct Proceedings of HCI98*, 1998.
4. R. Butterworth, A. Blandford and D. Duke. Using formal models to explore display based usability issues. *J. of Visual Languages and Computing*, 10:455–479, 1999.
5. R. Butterworth, A. Blandford and D. Duke. Demonstrating the cognitive plausibility of interactive system specifications, *FACS*, 12:237–259 2000.
6. M. Byrne and S. Bovair. A working memory model of a common procedural error. *Cognitive Science*, 21 (1):31–61, 1997.
7. P. Curzon and A. Blandford, Detecting Multiple Classes of User Errors, *Eng. for Human-Computer Interaction*, M. Little and L. Nigay (Eds) pp57–71, LNCS 2254, Springer 2001.
8. P. Curzon and A. Blandford, A User Model for Avoiding Design Induced Errors in Soft-Key Interactive Systems, *TPHOLs 2001: Supplementary Procs.*, R.J. Bolton and P.B. Jackson (eds), U. of Edinburgh, ED-INF-RR-0046, pp33–48, 2001.
9. R.E. Fields. *Analysis of erroneous actions in the design of critical systems.* PhD Thesis. U. of York, Dept. of Computer Science, Tech. Report YCST 2001/09. 2001.
10. M.J.C. Gordon and T.F. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher-Order Logic.* Cambridge University Press, U.K., 1993.
11. E. Hollnagel. *Cognitive Reliability & Error Analysis Method.* Elsevier 1998.
12. A. Newell. *Unified Theories of Cognition.* Harvard University Press, 1990.
13. D.A. Norman. *The Design of Everyday Things.* MIT Press 1998.
14. C.R. Roast. Modelling Unwanted Commitment in Information Artifacts, S. Chatty and P. Dewan (eds) *Eng. for Human-Computer Interaction*, pp77–90, Kluwer, 1998.
15. H. Thimbleby. Permissive User Interfaces, *International Journal of Human-Computer Studies*, (54)3:333–350, 2001.
16. H. Thimbleby, A. Blandford, P. Cairns, P. Curzon and M. Jones. User Interface Design as Systems Design. To appear in *the Proceedings of HCI 2002*, Sept. 2002.