Que sera sera — the problem of the future perfect in open and cooperative systems

Alan Dix

School of Computing and Mathematics
The University of Huddersfield
Queensgate, Huddersfield, HD1 3DH, UK
alan@zeus.hud.ac.uk

Abstract

When the pace of interaction with a task is too slow, the user's execution/evaluation loop is broken. Feedback normally says what *has happened*. However, when the task is slow, nothing has happened yet — the interest shifts to what *will have happened*. This poses two problems for the user. Firstly, recalling the appropriate context when a response eventually arrives. Secondly, maintaining the expectation that the response will come and when, so that appropriate action can be taken if it fails to materialise. The design question is how to support these activities, of which the latter has received little emphasis to date.

Keywords cooperative work, CSCW, delayed feedback, status/event, response time

1 Raison d'etre

A kitchen, somewhere in Cumbria, 7th November 1992, 8:37 am

Alan puts bread under the grill and waits for it to brown.

Email message received May 1993

```
From: forsyth  15:35:13  Wed 5 May 1993
Subject:
To: alan
  yes
```

2 Background

A previous paper (Dix, 1992) discusses the problem of pace in cooperative systems. That paper concerned the relationship between the pace of communication channels and the pace of the tasks performed cooperatively using those channels. The pace at which communications occur through a channel is the pace of interaction and is determined in part by the properties and intrinsic pace of the channel and in part by the pace of the task. Mismatches of pace lead to problems, both when the pace of interaction is too slow for the task, and when it is too fast. However, people are skilled in developing *coping strategies* which alleviate these problems.

This paper is related to the above work in that it concerns issues of time and pace of interaction. However, the focus here will be on the relationship between the pace of interaction (whether determined by communication channels or the task itself) and the pace of the users themselves. There are certain timescales which have qualitative differences for users. Two timescales will be important:

- *Short term memory* – From a cognitive viewpoint, we have a timescale determined by short term memory (Dix et al., 1993). Because of interference, this timescale is itself influenced by other tasks.
- *Task switching* – Either by choice, or because of interruptions, users may switch between tasks. This puts an upper time limit on the consecutive time spent on any particular task.

There is a large body of research on the effects of response time on user performance and behaviour (e.g., Shneiderman, 1984, Dix, 1987, Teal and Rudnicky, 1992). In most studies, the user is involved in a single task and is continually waiting for the system response. However, this paper will be primarily concerned with *very* long responses (from minutes to days) and the user will be expected to engage in other tasks between initiating an action and receiving a response.

3 The broken loop

Norman's execution/evaluation cycle has been an influential model of the way humans interact with their environment (Norman, 1988). Norman considers seven stages of interaction, but for the purposes of this paper, a simplified version, suffices (fig. 1.):

execution – Having established goals, the user decides what to do and does it. In addition, the user will have some expectation as to the effect these actions will have on the system.

evaluation – The user interprets the system's feedback in the light of the expectations generated as part of the execution stage. As a result of this goals may be modified and a new execution/evaluation cycle may begin.

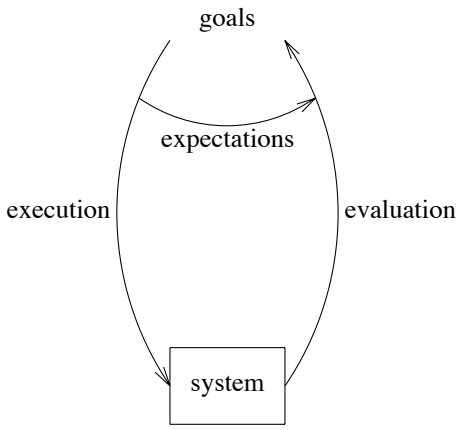


Figure 1: Norman execution/evaluation loop

Notice that in this description, the user is very proactive acting with intention to bring about some desired state of the system.

When the Norman model is applied to computer systems it is normally assumed that having completed the execution side the user then *waits* for the system feedback and then completes the evaluation stages. In such contexts, the system is assumed to be *closed* – the system is controlled by the user alone, with no interaction with other users or the

external environment. Where the response time is suitably fast, the whole cycle can take place within the user's short term memory span.

With cooperative or open[1] systems this simple application fails. Consider the sending and receiving of an email message. This may be one 'cycle' within a task yet may take hours, days or weeks to complete. Similarly, consider a factory. The pressure within a vessel is too low, so the operator turns on a pump, which will slowly increase the pressure. However, it may take minutes or hours for the vessel come up to the required pressure.

In these examples, the cycle is no longer completed within short term memory span — the loop is broken. In a closed system feedback tells the user about what *has happened*. In an open system, the emphasis changes to what *will have happened*: that a reply will eventually have been received, that the vessel will have reached its required pressure. The user knows *now* what to expect and yet may have to wait a substantial time before the desired result is obtained.

Arguably, models arising from industrial ergonomics are more appropriate here. These treat the user more as responding to stimuli from the system/environment: e.g., when the alarm rings, the operator turns off the pump. In a sense, this sees the loop as broken at the top (the user end) rather than at the bottom (the system). However, it radically changes the view of the user. The user is now *reactive* rather than proactive. This may often be the case in a large and complex system: the operator may not have a model of the system or any particular goals, but may be regarded as a cog within the whole. Perhaps a more acceptable way of looking at this would be within the remit of distributed cognition – the intentionality sits within the whole human/machine system rather than in the user per se. That is, one might argue that even when the user is acting purely reactively, one can see plans and intentions being acted out by the system as a whole.

Whether or not there is a larger view of intentionality, many actions are part of larger planned sequences. So, we still need to retain the essentials of the Norman loop, that action is purposeful, designed by the user to carry forward some aim or goal. In this case, the user faces two problems:

- *restoring context* – By the time a response comes, the user may have forgotten what it was about. For example, consider the email message which began this paper: "yes" – yes what?
- *maintaining expectations* – As well as remembering what should happen, the user must remember that it should happen at all. The user can no longer simply wait for feedback, but must remember that some response is expected in case a failure occurs and the response never comes.

Of these two issues, the first is reasonably well studied. In the groupware literature, we have examples such as Object Lens (Malone, 1987) where structured messages contain some indication of their source. At the level of social protocols, email users often include a copy of the senders message with their reply. In an industrial setting the fact that dials and alarms are associated with particular items of physical equipment acts as an element of context in its own right. However, the setting does not tell one *why* the event has taken place, the location may be a useful reminder, but the user still has to recall the rationale. Furthermore, in complex control rooms, where the alarms are no longer physically associated with devices, even establishing the locus of an event becomes a non-trivial task.

In contrast, the overhead of maintaining the expectation of a response is not widely recognised, perhaps because it is not explicit in the normal feedback loop. The expected feedback can be of two kinds: *event* or *status* (see also Dix et al., 1993, sect.9.4). Events, such as alarms or the receipt of an email message, can be assumed to be noticed when they arrive, but the user must take action if no event occurs within an acceptable time. Status feedback, such as a temperature dial, is more complicated. The required feedback is that the status eventually attains some desired value. The achievement of this value is an event

[1] That is systems which interact with the external environment.

in itself, a *status-change event*, but detecting that this event has occurred requires some positive action by the user. For both kinds of feedback an additional burden is placed on the user, but harder than simply restoring the context. In this case, the user has to recall *unprompted* that certain responses are expected to occur. Human memory is very efficient at recall from stimuli, but less so at this form of spontaneous recall.

Note that both the Norman loop and the reactive model of the user are silent on the second of these two problems. The explicit evaluation in the Norman model reminds one that context is required, but there is no similar placeholder for maintaining expectations – it is implicit in the closed loop. Similarly, the reactive view emphasises the user's formulation of a response to a stimulus and thus suggest that some context may be required to frame that response. However, a purely reactive model does not suggest actions when *no* stimulus occurs.

Note that in both open and cooperative systems, the user's own system is acting in a *mediating* role.[2]

Rather than working *on* the system, the user is working *through* the system to the world beyond. The feedback which we have been discussing above has been that of the 'world' (other users and physical processes), but one should not forget the immediate feedback from the local system. The user needs confirmation of an action at two levels (fig. 2):

- that the action has been received – from the system – fast
- that the action has had the desired effect – from the world – slow

In closed systems, the distinction between the two is often elided, but even there may be important (Dix et al. op cit.). Indeed, the whole direct manipulation style is built upon the hiding of the distinction. In contrast, the designer of an open system will need to offer feedback at both levels and will need to distinguish the two.

The rest of this paper will concentrate on the problem of maintaining expectations of responses as the problems of immediate feedback and restoration of context are relatively well understood.

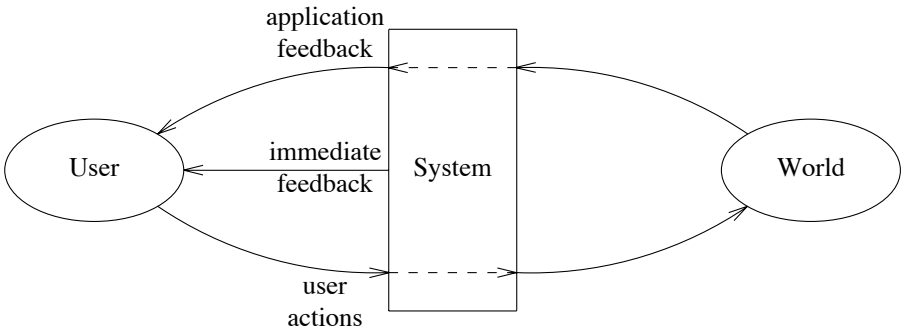


Figure 2: Mediated interaction

4 Solutions

The problem of delayed response is common in everyday life. We would therefore expect to find *coping strategies* adopted in various circumstances. This can lead to effective design guidelines as the system can offer support for the user's own, often effective, coping

[2]This is similar to Hollnagel's 'linked HCI' (Hollnagel, 1993)

strategies. In addition, there are additional solutions which are made possible by the presence of a support system. The fact that interaction is mediated by the user's local system means that it can record the user's actions on the world and process the responses. This has obvious implications for restoring context for the users, but can also help in the maintenance of expectation.

We can classify potential solutions by who or what takes *responsibility* for ensuring that the user's actions have their desired response: the user, the world (including other users) or the local system. We start with the responsibility firmly with the user – the aide-mémoire.

Aide-mémoire

From the knotted handkerchief to the Post-It note and 'to do' lists, people are well aware of their fallibility and put things in their environment to remind them. However, looking more closely, one finds that these methods are principally used for reminders that something has to be *done.* For recording expectations about others, people are far more sloppy. Office procedures are a classic example, being typically event driven. Your solicitor sends a letter on your behalf, but does nothing until either a reply arrives, or you ring up to pester. The better professionals will put some note in their diary of when the response is expected. Even where a diary or calendar is used, they themselves require 'monitoring' and significant events can be missed (Payne, 1993).

One can augment these normal forms of aide-mémoire with electronic equivalents. For example, the "Cc:" facility on an email system can be used to automatically post outward going mail into ones own mailbox. It can then be left there until a satisfactory reply arrives. The two problems with such techniques is the cost of putting in the information and the noise from having too many reminders around.

As a design concept, the aide-mémoire can be applied in other contexts. For example, consider again the example of the operator wishing to increase the pressure in a vessel. Imagine adding a small red needle to the pressure dial, which can be set by the operator. When the operator opens the valve the dial is also set to the desired pressure. Then, when the operator next scans the control panel, any dial where the needle value and the actual value differ demand attention. Of course, the difference may be acceptable as sufficient time has not elapsed in which case the operator can simply ignore the discrepancy until next time the dials are inspected.

Note how the use of an aide-mémoire shifts the burden from maintaining expectation to restoring context. The fact that the aide-mémoire is there reminds one that *something* should happen (or be done), which then means that one has to remember *what* it is. In the case of the needle on the dial, the aide-mémoire is part of the items to which it refers, the only remaining job for the operator is to recall whether the target pressure should have been reached yet.

There are problems with aides-mémoire, consider an extreme case. One year you forget your sister's birthday, so you tie a knot in your handkerchief so that you don't forget next year also. The disadvantages are obvious – a (very dirty) handkerchief with 73 knots and you can't remember what any of them were for.

When considering system support, the characteristic feature of an aide-mémoire is that the information it captures is *uninterpreted* it is the user who assigns meaning to it. However, the electronic domain can add value, for example, the user may be also to link a reminder to the screens/applications required for interpreting it.

Trust and predictability

The reason that many office procedures are event driven is that this deliberately removes the need to maintain expectations. The individual can perform a task, send results or requests to someone else, and then reach closure, not considering the task again until some further stimulus (usually piece of paper!) arrives. Such systems are built upon *trust:* having asked someone to do something, one trusts that they will perform the action,

having requested information, one trusts that the reply will be received. Note that this effectively shifts the burden of responsibility from the sender to the recipient.

Predictability of physical and electronic systems has a similar effect. If one knows that increasing the inflow to a vessel by 10% *always* increases the pressure by 5% within 20 minutes, there is no need to monitor the process.

In some sense trust is a particular form of predictability. However, shifting responsibility to another individual has different implications to shifting it to a machine or process. In the former case, one has someone else to blame when things go wrong! Of course, things do go wrong, especially in an office situation where the chains may become quite long, often crossing corporate boundaries.

In control engineering terms, using predictability allows one to swap closed loop control for open loop control. The problem of delayed feedback ceases to be a problem because feedback is no longer required – when something is asked to be done, one assumes it will be done. However, open loop control is inherently more fragile and demands total knowledge and trust. So, because things don't *always* happen as expected, even where open loop control is used, it will often be enclosed within some wider procedure where feedback in monitored.

System support here would be ways of annotating requests to make the shift of responsibility more obvious. This is particularly a problem with email communications where different corporate (and individual) cultures put a different emphasis on the 'strength' of messages. In some email cultures, ignoring email is allowed in a way that ignoring a paper communication would not be. A system like Coordinator (Winograd and Flores, 1986) at least made these mutual expectations clear.

Tell the system

Of course the system can do more to help if it knows what the user's expectations are. As the system mediates the feedback from the world, the user could explicitly tell it when certain responses are expected. For example, when sending and email message, one could have a 'reply by' field, which need not be sent to the recipient (but may be). This system could then alert the user if no reply is received by the specified time. The same technique can be applied to industrial situations, for example, telling the system that vessel A should be at an operating pressure of 4 bar within 20 minutes. The advantage is that the *responsibility* has now moved entirely to the system. If the reply is not received or the vessel does not come up to pressure, the system alerts the user. Once the user has told the system nothing needs to be done until some response or alert is received — a form of non-military 'fire and forget'.

The problem with such *explicit* passing of responsibility is the effort it places on the users, both in terms of the time taken to enter the data and the complexity of externalising their expectations. For large scale tasks this may not be a significant barrier, especially if the applications are closely linked to familiar tools such as diaries or PERT charts where the significant times can be entered. However, as the pace of activity increases, the load becomes unacceptable. Even for email communication, with rates commonly between 20 and 100 messages a day, explicitly setting 'reply by' dates is likely to be taxing.

An alternative is to make this passing of responsibility *implicit.* This can be achieved by system prediction followed by user confirmation. Various office tools already use predictive features as task accelerators, for example, the latest versions of Microsoft Excel have intelligent sum operators which guess what you want to sum. In a similar way, an email system could guess the expected reply date based on the normal rate of interchange with the recipient. This could then be used as a default for the 'reply by' field. Alternatively, user defined rules could be used in a structured message system to determine the reply time as a function of the recipient, message type etc. The important thing is that such techniques accelerate the process and require only confirmation (which may even be silence).

In an industrial setting predictive interfaces are already used, but with an emphasis on planning. The operator enters planned actions and the system shows the projected

changes based on some model of the physical process. Given the system has supplied predictions it is not unreasonable to expect it check those predictions and inform the user when the system falls outside those parameters.

It is interesting to note that if predictive systems took on the responsibility to alert users, they would become more honest. Imagine a colleague gives you some information upon which you act. Later your colleague becomes aware that information is wrong. You would be rightly aggrieved if your colleague did not inform you. This is precisely the situation with most predictive interfaces!

5 Two case studies

We have now seen several ways of dealing with the problem of delayed feedback. We will now look at what these tell us about two specific scenarios.

Air traffic control

Air traffic control is a complex activity. It involves coordination between the controllers within at the ATC centre and between the controller and the aircrew. Furthermore, it involves the physical movement of aircraft. That is it is open *both* by virtue of cooperative activity and physical processes. One critical part of the ATC process is the use of flight strips, small strips of coloured paper referring to individual planes (fig. 3). On this is printed and written various information, some constant (usually) such as the destination, and some variable such as the flight level (height).

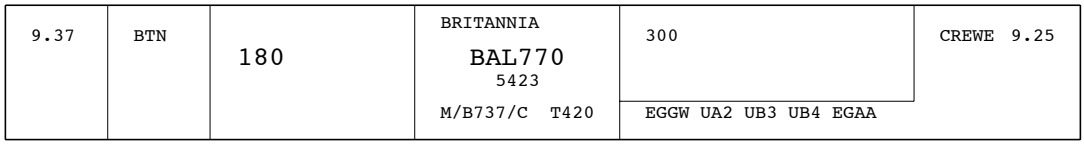


Figure 3: Air traffic control flight strip

When the controller asks the aircrew to take the aircraft to a new flight level there is a delay before that height is reached, due to the communication and to the physical behaviour of the aircraft. Furthermore, the controller's attention cannot remain with that aircraft, but must shift to the many other tasks within the air sector. One would therefore expect to find some form of aide-mémoire or other means to cope with the delayed feedback, and the obvious locus for such information is the flight strip.

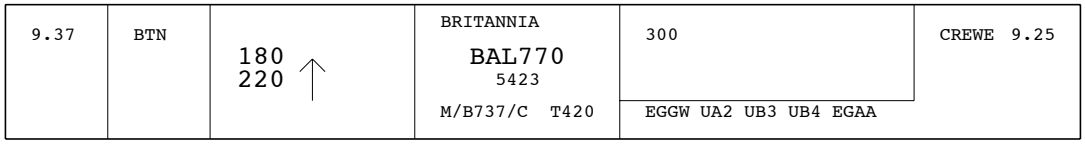
As predicted the flight strip is used in precisely this way:

> "When a controller gives an instruction to a pilot, for example to ascend to flight level 220, s/he marks this on the strip: in this case, with an upwards arrow and the number 220 (*fig. 4.i*). When the pilot acknowledges the instruction, the controller crosses through the old flight level on the strip (*fig. 4.ii*). When the new height is attained, the controller marks a check beside it (*fig. 4.iii*)."
> (From Hughes et al., 1992 figure references added).

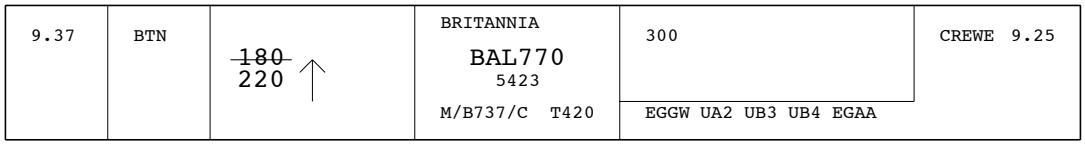
Note that the flight strip acts as an aide-mémoire first for the completion of the communication task and then for the completion of the whole task. A quick scan of the flight strips (probably not even conscious) will reveal all those with two flight levels (indicating that confirmation is still awaited) or those with unchecked flight levels (confirmed, but not attained).

Clearly any system to automate flight strips must either replicate this facility or produce some equivalent one. For example, if flight strips were on-line then the system could

(i) Controller gives instruction to a pilot "ascend to flight level 220"



(ii) Pilot acknowledges the instruction



(iii) New height is attained

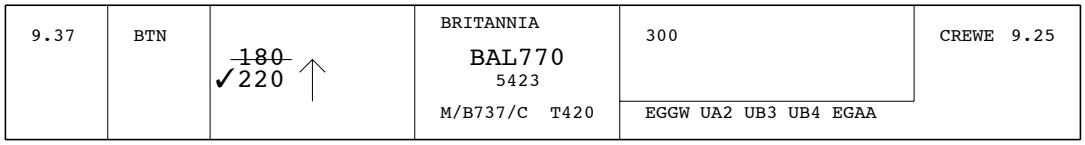


Figure 4: Flight strip is annotated to record expectations

automatically check when aircraft did not reach their desired height. The choice between such alternatives is not simple as a single mechanism may fulfill several roles, but the analysis both points one towards the existing mechanisms and gives one a repertoire of alternatives.

Car radio tuning

We now look at a problem which arose during the design of a new car radio. The interface to the radio is located at the front of the car and has the familiar tuning buttons and frequency display. A user interface micro-processor handles this interface. However, the actual radio tuner is controlled by a separate micro-processor in a different part of the car. The two micro-processors communicate via a (not too fast) data bus. The data bus introduces delays and thus we are considering a mediated interaction as described earlier. The timescales involved are actually quite short, possibly a second or two at very worst, probably only a fraction of a second. However, the driver is (one hopes) involved in a variety of other, more demanding tasks, such as keeping the car on the road. Thus even these comparatively short timescales lead to a broken execution/evaluation loop.

The specific design problem faced was the behaviour when the user pressed the tuning buttons. The interface controller would send a message to the tuner asking it to retune. Two options for the feedback were initially considered:

i) Display the new frequency immediately the user pressed the button

ii) Wait until the tuner responds that it has retuned

The rationale for the former is that the user ought to gain immediate feedback. the justification for the latter is that the display ought always to reflect the actual state of the system.

In fact, this is a highly complex situation. There are two feedback paths to the user: the frequency display and the sound of the radio itself. These two are usually in agreement, and the driver will be unaware of the distributed nature of the cars computing architecture, so when discrepancies arise there is liable to be some confusion. In addition, the visual feedback has two functions. On the one hand it gives feedback on events: "your button press has been noted". On the other hand it gives a continuous status display: "the channel you are listening to is 92.5MHz".

From our previous analysis of mediated interaction we see that feedback is required at two levels:

- interface – that the user's button press has been registered
- application – that the channel has actually changed

Option (ii) gives *no* immediate interface feedback. If the network lags are short enough this may not matter, but anything more than a few hundred milliseconds is unacceptable for button press feedback. The result is likely to be that the user will repeat the button press, leading to an overshoot of the desired station and 'hunting' behaviour. In fact, this scenario is more likely to affect the passenger as the driver may operate the radio by touch whilst keeping attention on the road. Tactile feedback from the buttons is thus likely to dominate for the driver.

Application feedback comes through the sound of the radio in both cases. The justification for option (ii) says that there should be a status–status mapping between the state of the radio and the frequency displayed. However, where a status–status mapping is mediated by messages there is inevitably some discrepancy. In option (i) the frequency display would be ahead of the actual tuned frequency, but option (ii) suffers as the display would lag behind. Thus there will always be some period of discrepancy and if this is not to cause confusion some means of flagging it should be supplied.

We now have two requirements: to show immediate feedback and to indicate discrepancy in the displayed frequency. We can now consider a third option:

iii) Some instant display change to an 'undetermined' state – that is some display which indicates that change is in progress. There are various ways of showing this:

iiia) spinning dials – The digits of the display quickly alter as if they were searching for the new frequency. This would have the advantage of giving an immediate sense of change, but would mean that there would be no feedback of where that change is going. Thus if the user hit the wrong key (tune down rather than up), there would be no feedback until the system had stabilised.

iiib) flash the number – either the new number (answering the criticism of (iiia), or alternate between old and new. Possibly this would be too distracting.

iiic) busy indicator – Have an additional busy indicator so that the user knows that the action is not complete.

Like the ATC case study, the final choice of solutions depends on a variety of factors (e.g., must use standard radio display screen), the essential feature is that the analysis highlights the central elements of the design.

6 Summary

We have seen how in open and cooperative systems there is often a lag between a user's action and the associated feedback. This breaking of the traditional execution/evaluation loop puts extra burdens on the user: to recall the context *when* a response arrives, to maintain the expectation that an response *should* arrive. We looked at four potential solutions to the latter problem:

- Aide-mémoire
- Trust and predictability
- Explicitly telling the system – passing on responsibility
- Implicitly telling the system by confirmation of system predictions

We can analyse existing situations to uncover uses of these solutions and also use them to generate design alternatives. The former is important as otherwise existing coping strategies may be spoilt by the introduction of new support systems.

Two case studies have revealed how these problems occur in practice and demonstrate some of the resulting design space. In both cases, the analysis suggests various alternatives, but the choice depends on other factors. The importance of the analysis is that from seeing that a phenomena or problem exists, we can go to its underlying cause and thus open up the possibilities for solution.

Acknowledgements

This work was funded by SERC Advanced Fellowship B/89/ITA/220 and SERC grant GR/F/01895. Thanks to the anonymous referees who gave several pointers to related work and to those who have listened and commented on earlier verbal presentations of this material.

Raison d'etre (ctd.)

A kitchen, somewhere in Cumbria, 7th November 1992, 8:39 am

Alan awakened from reverie by the smell of burning toast

References

Dix 1987 A. J. Dix, "The myth of the infinitely fast machine", pages 215–228 in *Proceedings of HCI'87: People and Computers III*, D. Diaper and R. Winder (eds.), Cambridge University Press, 1987. Also in A. J. Dix, *Formal Methods for Interactive Systems*, Academic Press, 1991.

Dix 1992 A. J. Dix, "Pace and interaction", pages 193–207 in *Proceedings of HCI'92: People and Computers VII*, A. Monk, D. Diaper and M. Harrison (eds.), Cambridge University Press, 1992.

Dix 1993 A. Dix, J. Finlay, G. Abowd and R. Beale, *Human–Computer Interaction*, Prentice Hall, 1993.

Hollnagel 1993 E. Hollnagel, "The design of reliable HCI: the hunt for hidden assumptions", pages 3–15 in *People and Computers VIII: Proceedings of the HCI'93 Conference*, L. Alty, D. Diaper and S. Guest (eds.), Cambridge University Press, 1993.

Hughes 1992 J. A. Hughes, D. Randell and D. Shapiro, "Faltering from ethnography to design", pages 115–122 in *CSCW'92 Proceedings*, J. Turner and R. Kraut (eds.), ACM Press, 1992.

Malone 1987 T.W. Malone, K.R. Grant, K. Lai, R. Rao and D. Rosenblitt, Semistructured messages are surprisingly useful for computer supported coordination, *ACM Transactions on Office Information Systems*, 5(2), pp. 115–131, 1987.

Norman 1988 D. A. Norman, *The Psychology of Everyday Things*, Basic Books, 1988.

Payne 1993 S. J. Payne, Understanding Calendar Use, *Human–Computer Interaction*, 8(2), pp. 83–100, 1993.

Shneiderman 1984 B. Shneiderman, Response time and display rate in human performance with computers *ACM Computing Surveys*, 16(3), pp. 265–286, September 1984.

Teal 1992 S. L. Teal and A. I. Rudnicky, "A performance model of system delay and user strategy selection", pages 295–305 in *CHI'92 Proceedings*, P. Bauersfield, J. Bennett and G. Lynch (eds.), ACM Press, 1992.

Winograd 1986 T. Winograd and F. Flores, *Understanding computers and cognition : a new foundation for design*, Addison-Wesley, 1986.