

Designing for humans

Professor Harold Thimbleby
Bernhard's Inn, Gresham College, London
with Dr Ann Blandford, University College London

13 February 2003

Pilot error. Human error. We often hear about human error when things go wrong. We often blame ourselves when we have problems. This lecture will show that certain sorts of human error are persistent and predictable; when problems arise from these causes, the errors are design problems, not (or not entirely) 'pilot error' or a user's fault. As well as discussing the underlying causes, and showing some examples, the lecture gave some solutions.

When people interact with devices and gadgets, including computer systems, they generally have some objective — something they want to achieve, to have fun playing a game, or to buy a return ticket to Blackpool, or to learn something about the tourist attractions in Blackpool, whatever. In many cases, the actual outcome of the interaction is different from the objective. The user may achieve less than intended, or perhaps they may achieve more than intended — maybe they will come across a two-for-one offer on tickets to Blackpool, so they'll only have to pay half what they were expecting! In many cases, the reason for achieving less can be traced back to something specific going wrong in the interaction, such as the user acquiring incorrect beliefs about the system and acting on them, or the user forgetting to complete some key step in the interaction.

If, as an independent observer, you watch the user's interaction you can account for the problem in terms of the behaviour of the user and the system. If you know what's meant to happen, you can spot when something goes awry. However, observation merely gives a surface-level account, which may miss deeper causes. For example, did that executive grab his ticket and leave his change behind, because he decided missing the train would be a worse outcome than losing the change, or because he forgot and was flustered?

The point of this lecture was to argue that we can deal with these issues; devices such as ticket machines can be improved. It is possible to change designs to reduce the impact of problems such as 'forgetting' — even though some problems, like deliberate decisions to leave change behind, cannot be dealt with.

Marketing has led us to think that people want usability. This is misleading, and is one reason why designers do not anticipate and handle user error adequately. User error is a very important source of usability difficulties. There is a view of usability that it is concerned with users having pleasure and achieving interesting or useful effects; but the first concerns of usability should be to avoid things going wrong, whether that is considered systems failure or user error. In this sense, usability is a privative — that is, something that is only detectable by its absence (privative is a philosophical term used by Aristotle). Like darkness (the absence of light), comfort (the absence of irritation) or vacuum (the absence of matter), usability is the absence of problems. An understanding of errors and their causes is an important step towards developing more usable systems. Hence we should design to reduce the risk of errors (or, to be more precise, the costs of accidents that result from probable errors). Let us accept that errors happen but, once we understand the causes of particular classes of error, we are in a good position to try and eliminate those errors — and this will create usability. Feeling good is not sufficient.

Although errors may appear to be random and unpredictable, if we start classifying them according to their likely causes, we find that similar sorts of error crop up in apparently unconnected places. For example, some photocopier designs are prone to the user walking away with their copies but leaving the original behind on the glass, or forgetting their photocopy card and leaving it behind. The 'forgetting' error here shares key features with the ticket machine error of forgetting change. In both cases, the user has a main goal to achieve — getting copies, or getting a ticket — but has to do something extra — getting the original back, retrieving the card from the card-reader, collecting change in the slot — before the wider goal can be achieved. And if they get distracted or aren't concentrating, they are prone to forgetting to complete all parts of their goal. This is a common problem that arises in various situations.

In some contexts, solutions to the problem have been found. When withdrawing cash from an ATM [cash machine], users are now required by the design of the machine to remove their bank card before any money is delivered, so that achieving the main goal is the last thing the user does. Another example: in most supermarkets, supervisors have a key, attached to their belt, that they have to use in the till before they can override till settings; this mechanical design avoids them walking away leaving the till insecure.

We start to see common patterns in these design solutions, and we can devise design solutions that can be transferred from one situation to another. The lecture will give more examples and solutions. The type of error just described is sometimes referred to as a post-completion error — there are small things left to do when the interaction finishes, and the outcome is sub-optimal for user, system or both. While it is possible to create a reasonably complete taxonomy of behavioural errors at the level of 'add an extra action,' 'omit action,' 'transpose actions,' 'replace one action with another,' to the best of our knowledge no one has generated a complete taxonomy of errors in terms of their underlying causes. In this lecture, we will be considering a representative range of causes, showing how they crop up in a variety of contexts, with a view to building up a repertoire of design solutions that can be applied to causes once identified. Thus, in many cases designers will be able to anticipate, and therefore avoid, some errors before they arise. Often, when you make mistakes you make certain, pretty predictable, sorts of mistake (including post-completion errors); good designers know this and can

help you avoid the problems. In a strong sense, then, in these cases it is not you who makes mistakes but the designer.

Another error that might, in some situations, be classed as a post-completion error is that of the user of a VCR who takes quite some time (and mental effort) to program the timer ... but then forgets to put the VCR into 'record' mode. If we view 'record' mode as the normal state, which the user has had to unset in order to do the programming, failure to reset it back into record mode is a post-completion error. There is, however, another plausible explanation of this error, which is more closely aligned to the way people communicate with each other: we generally communicate efficiently, not saying more than needs to be said. If you turn up at Euston and say "Blackpool?" to a member of station staff, you will almost certainly be told what time the next train departs and which platform from. Because of the context, a minimal question can expect an appropriate response. Similarly, since the purpose of a VCR is to record and play back videos, it is often natural to assume that entering all the required information (times, channel, etc) is sufficient to 'tell' the VCR what to do next. Again, the problem can be found in other situations, such as the railway ticket machine, which demands that the user confirm their purchase before inserting payment (rather than assuming confirmation because the user is choosing to pay). We have observed bemused rail travellers walking away from the ticket machine at this point, assuming that the machine is not accepting payment.

We can enumerate, and describe with reasonable precision, various other classes of errors. Capture errors occur when two sequences of actions are very similar, but one is much more familiar than the other — for example, if you see someone 'indicating' with windscreen wipers, there's a pretty good chance that they are much more familiar with driving a car which has these stalks on the opposite sides of the steering wheel. Blind alleys occur when the user doesn't have enough information to know whether or not their objective is achievable, and has to find out — for example, phoning someone to get information from them, not knowing whether they are available or have the information. The outcome of such an interaction may be satisfactory, or may be a blind alley, where the user has gained some information (e.g., about the non-whereabouts of the person), but hasn't actually achieved their objective. Finally, interaction traps occur when the system state and the user's knowledge of that state get out of step, such that the user acquires incorrect beliefs about the achievability of their objective. For example, they may believe something is unachievable when actually it can be achieved, or vice versa.

One of the authors fell into an interaction trap when checking information about this lecture. Requesting information about future events on the web site, she changed the default search period from being today until the end of the year to being today until the end of April, only to be informed that "This date is not valid 04/31/03." Why is this objective not achievable? Of course: because April only has 30 days. Is this a case of foolish user or poor system design? It is certainly a case of avoidable problem: the web site could be redesigned. For Gresham Lectures this isn't an exciting issue, maybe, but if this was a medical system or something in an aircraft cockpit the same sorts of predictable design problem would be very serious, and almost certainly raise liability problems.

The Designing for Humans lecture reviewed these issues and their causes (post-completion errors, capture errors, etc), and gave examples (using some simulated systems). The lecture showed how careful design can lead to more effective systems in many areas of life.

These issues are all human factors, and contrast with the content of Thimbleby's preceding lectures (Designing Gadgets, 26 September 2002; Designing Microwave Cookers, 24 October 2002; Designing Mobile Phones, 28 November 2002). To make the Designing for Humans lecture more exciting, it was run as a debate, with motion "This hall believes that training in maths is more important than human factors for designers." Harold Thimbleby proposed the motion, relying on his arguments from the previous lecture; he spoke for only a few minutes. Ann Blandford opposed it, and covered the points listed above in these lecture notes.

The Gresham audiences include users, designers and manufacturers who will want to express views about product design! Before the debate, the votes were 7 (and an on-the-fence 0.5) for, and everyone else against.

In their summaries, Thimbleby argued that maths makes systems easier to use, more reliable and helps them work. Any other approach is hindsight, and comes too late for almost all sorts of design. Blandford, arguing against the motion, summarised by arguing that people are different but there are common features beyond the really obvious; we can design with or against those human factors, and it is important to understand them.

After the debate, the votes were 13 for, 24 against, but the debate was nicely drawn to a close by a suggestion from the audience: that the motion was 'rigged' to be interesting, but had the motion been "this hall believes designers should be trained in both maths and human factors" then everybody would have assented. The lecturers wholeheartedly agreed — but maybe the debate would have been less exciting!