Creating discerning users

Harold Thimbleby

Middlesex University
Bounds Green Road
London
GB
harold@mdx.ac.uk

ABSTRACT As technology progresses, we are still surrounded by complex and difficult-to-use systems. Usability engineering appears to have had little impact on practice. Why is this, and what can be done to contribute to improving system design? This paper surveys the pervasive culture of unusability, and proposes that HCI should promote public understanding of science. A particular proposal is also put forward: for an interactive web-available "HCI workbench," an interactive design assistant, that can be used by consumers, collaboratively both researchers and practising designers.

KEYWORDS "Unusability Culture," Interactive Design Assistant, Public Understanding of Science

PAPER #11683

1. INTRODUCTION

Video recorders, aircraft flight decks, and desk top computers are all examples of complex interactive systems. Interactive systems are central to modern life, but there is widespread dissatisfaction with even 'simple' systems: they are too often awkward and frustrating to use. Poor design, including poor user manuals, have been blamed for accidents — about 90% of all computer-related deaths are caused by poor user interface design (MacKenzie, 1996). Despite the 'obvious' bad design, there is widespread acceptance for the problems, both from users and professionals.

This paper describes the culture accepting usability — or lack of it — as our own responsibility. It then makes a positive suggestion: to promote public understanding of science, particularly through good demonstration systems, with good user interfaces.

(A companion paper, also submitted to this conference, describes how this may be done in more technical detail. However, both papers stand alone and can be read independently. Some paragraphs are shared between the papers in case only one of them is accepted — or if they are reviewed independently; the duplication will be removed if both are accepted.)

2. UNUSABILITY CULTURE

Although unusabilty is bad in almost every situation, users seem to accept it, and manufacturers continue to make unusable systems. Why is there this unusability culture, and why is it so persistent?

Human society is complex, and although there is no *a priori* reason why society should be complex, we are embedded within it and in principle cannot understand it reliably. Society in fact is at the limits of human abilities to comprehend. We could argue, for instance, that our educational system has a hard job succeeding, and there are many failures — not least because our technology is so complex. Indeed,

modern technology is made by people who have succeeded through a long period of education, and stand, as it were, at the pinnacle of the educational pyramid. The few sufficiently-skilled people design and build computer systems, and almost inevitably, because they do the best they can, what they do is hard to comprehend for the many other people less skilled than they.

Exacerbating the complexity: the people who use systems are themselves in society, and therefore push their own limits. Businesses are competitive and try to out-do each other — hence relying on exploiting skills they hope other businesses cannot use so well. Thus computers-in-use present an even more complex situation than the already complex-enough society.

Complex systems are hard to describe. That user manuals are often wrong in detail suggests that manufacturers even find their own products hard to describe! If we can't say exactly what a video recorder or a word processor does, how are we going to improve it?

The following sections (§3, 4) by no means cover the full range of issues. Other relevant discussions are Norman (1998) and Thimbleby (1990a, 1990b, 1992, 1993, 1996, 1998a). A range of non-HCI issues are covered in: Christensen (1997) which provides a business perspective — don't listen to customers; Paulos (1990) is a mathematical perspective — users and designers are innumerate; Piatelli-Palmarini (1994) is a psychological perspective — none of us make sensible decisions, either to design or to consume; and Tenner (1996) provides a pessimistic "any advance has problems" perspective!

3. CULTURE AFFECTS USERS

3.1. HCI commodified

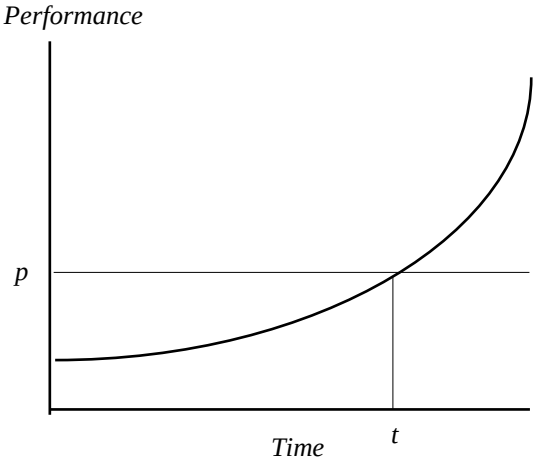
Many user interfaces, about which HCI has a proper concern, are also consumer items. We buy word processors, email systems, video recorders, and so on. Thus in many user interfaces, we have an actual stake in the outcome of interaction. Few people would be happy admitting they made a financial mistake buying product X over product Y. Moreover, reducing cognitive dissonance means that people will be tempted to rationalise any difficulties they do have! Thus any branch of HCI that takes as its starting point the possibility, the desirability, of changing the *status quo* has a deeply embedded cultural hurdle to overcome.

In everyday life, when HCI problems are experienced, we have been trained to take responsibility for those problems. We can buy help books, or computer upgrades, or the market will offer new technologies providing new features and 'solutions.' In every case, fixing HCI problems has practically become a consumer pastime, though disguised as buying into better systems — as fashion, in fact. Behind the fashion is the cultural view that usability problems are the user's problems, not the designer's. As user problems, the user can solve them by buying a 'better' system, or better documentation, or 'upgrading' their current one.

In all cases, the responsibility for action is directed *away from* improving HCI.

3.2. Delaying quality

Norman (1998) credits Christensen (1997) for another view of "avoiding HCI" that has become endemic. Technology is getting better, and has increasing performance over time. We can represent this by the graph, below, using a line of positive slope (it is not necessary for our purposes to worry about the precise shape of the line, or exactly what 'performance' is measuring). For any particular task the user has, some level of performance *p* will be required. From the graph it is clear that there is a threshold point when the lines intersect, at performance=*p* and time=*t*. Before *t*, technology is delivering inadequate performance; after *t*, technology can deliver more than adequate performance.



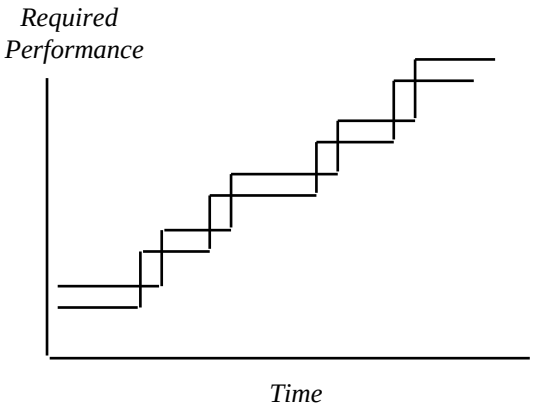
So before the threshold time, all manufactures have to do is promise increased technical performance (which is easy, since technology is getting better all the time). After time *t*, products get distinguished not by their performance but by more subtle — and harder to supply — properties like usability. It is therefore in manufacturers' interest to increase *p*, because this will postpone the threshold time. For technologies like wrist watches, we are long past the threshold, and they are now fashion

items. For many interactive systems, like word processors, we should also be well beyond the threshold. So why aren't word processors much better?

3.3. The tragedy of feature stepping

The tragedy of the commons is that farmers acting in their own best interests over-graze common land. If the land is being over grazed, the community should hold back. But it is to the advantage of any individual to graze a few more of their own animals, especially if other people are removing theirs from the common land. The same effect occurs with technologies that are shared: each person benefits by having the "best," but because not everyone can simultaneously have the best, a feature-stepping race occurs. The result is that few people ever benefit from their investment. Systems and training becomes obsolete. (The millions of tons of computers the UK landfills annually is a testament to the continual over-taking of once-leading computers.)

What we call "feature-stepping" is illustrated in the diagram below. For simplicity, assume there are just two users. One user has some required level of performance. They get a message from the other user, who is using a more advanced system, and so they are forced to upgrade. However, by the time they upgrade, the manufacturers have improved the performance of the technology, so they upgrade *beyond* the level of the other user. Then the situation between the users is reversed, and the first user wants to upgrade … and so on. Each user upgrades alternately, and if the manufacturers play the game properly, the rate of performance requirements increasing due to stepping stays above the performance the technology can deliver. Thus manufacturers can keep users permanently behind the threshold time.



Manufacturers are aware of these community pressures and they may take further steps to encourage the "upgrade habit." Users of old version systems are obviously encouraged to upgrade so they can read later versions; but they may be forced to upgrade by their "richer" colleagues because later versions of the software may deliberately not be able to read older versions. In other words, both old and new users want old users to upgrade. This is a powerful way of increasing *p*, especially when many users are spread around the world, and there is no way (e.g., organisational purchase controls) of controlling their understandable urges to keep up.

Would standardisation help, so all users and manufacturers had an agreed level of performance for tasks? Possibly not. Technologies like the web are a good example of this behaviour, but with the twist that standards are set that anticipate future technologies. Naturally the World Wide Web Consortium sets standards that are the best possible — but that means they are above what most users are capable of. Thus the stepping is always above the performance available that technology delivers, and users are permanently kept hoping for the future threshold.

This seems somewhat cynical, but Microsoft (as an example of a leading manufacturer) have admitted doing it (Gibbs, 1997): they are quoted as saying, "if we hadn't brought your processor to its knees, why else would you get a new one?" This looks like stepping, pitting harware and software against each other. Once a user buys the new processor, they are in a position to put pressure on other users to upgrade to keep up with them. It is interesting that most processor purchases include bundled software: thus giving users what seems like a free upgrade. Actually, it's marginal for them, and expensive for everyone else who they cause to step!

Later versions of Microsoft Word cannot read files saved by certain earlier versions. Thus users are faced with the choice of upgrading or being isolated — or the other user might try downgrading (and discover that de-installing has been made surprisingly tricky).

People who use the latest standards force "backward" users to upgrade. They then upgrade — but they buy into technology that is the latest, and therefore ahead of everyone else. So the co-stepping cycle goes. (Like the tragedy of the commons, each individual's sensible behaviour is to the whole community's detriment.)

In short, consumers of complex computer systems have been kept — for marketing reasons and

so on — to the left of the threshold. Kept "in their place" their job is to consume, rather than to demand better systems, with better HCI. Nobody is critical of bad HCI, because their systems are anyway inadequate and need upgrading …

3.4. Homeostasis in effort

As technical performance improves, society's standards for tasks will also be pushed upwards, if we assume there is a constant social value associated with the costs of performing tasks. Thus as things become easier to do, the social value of what users achieve decreases. To maintain the same social value, then users must do more sophisticated things.

Word processors illustrate this phenomenon well. Once it would have been sufficient to print text that looked written by a typewriter, but as this became easier to do, more fonts were required, then colour, then clip-art, and so on. Thus p is increased, and becomes an upward curve above the technological performance. The consequence, again, is that the time to the threshold of good HCI is postponed.

3.5. Consumer (in)action

In the 1960s, some cars were badly designed and unsafe to drive. As Ralph Nader exposed (Thimbleby, 1993), the prevailing cultural assumption was that drivers had accidents, and therefore drivers were responsible for the behaviour of cars. For example, if a parked car rolled down a hill, the driver should have applied the parking brake properly, they should have turned the wheels so the car would roll onto the kerb, and so on. That the car might have been badly engineered and have a feeble parking brake was thereby disguised.

The same sort of problem arises with HCI. Users are persuaded that their problems are their own problems, and that to use complex software they should take responsibility for their own training. Users have usability problems, *so* users should learn how to use computers properly.

The computer-based society is certainly a complex place, and people do need to be computer-literate because that is how the world is. But this practical response must not be used as an excuse to make systems more complex than they need be, because users will take it upon themselves to learn how to use them. Obviously something has to be known about computers and some training is appropriate; but at present the balance is clearly in manufacturers' favour. Indeed, manufacturers often commodify their learning material, thus making further profit by providing systems that require additional training! Companies like Microsoft have sophisticated certification processes, not only earning money, but creating a culture of dependency on them.

The cultural blindness may have little to do with computers or their perceived mystique. It may be a natural response to complexity: some people like using complexity as a subtle means of taking advantage over others less knowledgeable about the rules. For example, the UK tax system was too complex, and the Inland Revenue itself could not cope. Rather than simplify the system so that more people could cope with it, a law was passed effectively to require people to calculate their own tax. Thus nothing was simplified, but with the backing of the law, the country's population had to take it on themselves to be responsible for understanding an unnecessarily complex system.

Another example, backed by European directives, is just as computer technology would have permitted the introduction of simple user interfaces to car management systems, it has been outlawed. The reasoning is that only manufacturers should be able to adjust emission settings. Of course this is spurious; the consequence is that what could have been trivial for everyone has now become impossible for all but approved professionals.

3.6. Complexity suits manufacturers

Complex interactive systems are hard to understand. It follows that when a potential user is selecting between several systems (e.g., in a shop) they will be unable to make a rational choice based on usability and task-fitness. Instead, they will choose on brand name, appearance, price, and other superficial factors (such as colour). Thus, by making systems complex, manufacturers can push users into making decisions on much simpler factors, and purchasing patterns will become more predictable.

Which is better for a company: to sell a proportion of their systems, proportionally to distribution in the market, or to risk selling none because users can tell the system is inappropriate for some purposes? A cautious manufacturer might wish to hide possible problems — at least until the purchasing choice has been made.

In short, complexity — bad HCI — suits manufacturers, certainly for consumer products, and probably for products that are purchased in more formal procurement procedures (even if there are safety critical criteria).

3.7. Drama

Interactive systems, from video recorders to aeroplane flight decks, are highly complex, but they can be made to look deceptively simple by 'good' industrial design. The consequence is that problems only become apparent during a crisis. When a video recorder manual is lost and the programme to record starts *now*, or — more worryingly — when an aeroplane pilot is under pressure, mistakes are made. Often these mistakes can be traced to poor interaction design, including bad manuals, and poor quality control. Bad manuals, for example, are common because the technical authors are rarely in a position to adequately understand the systems. The problem is that computer systems can be made to look good enough for a demonstration or even to pass a test (Thimbleby, 1996), but without proper theories and knowledge of the precise nature of the systems, both informal and formal assessments reveal very little about the full range of possible behaviours of the systems. Indeed, consumer products often have cosmetic features and demonstration modes to entrance users, and hence delay the discovery of problems.

3.8. The Year 2000 bug

The Year 2000 bug is arguably the largest single problem ever facing humanity; it is certainly the most expensive problem of our own making! It arises quite simply from unprofessional practice by system manufacturers. The poor practice includes bad programming, bad design, bad diagnostic, bad documentation, and refusal to take responsibility.

The lack of care for users implicit in software warranties is well known (Thimbleby, 1990a), but the Y2k bug makes the culture of "usability is the user's problem" stark. Manufacturers charge customers for fixing problems, entirely of the manufacturer's own making. In some cases, this charging occurs several times as the manufacturers repeatedly upgrade their badly-engineered systems.

Insurance companies, the Halifax Building Society being a good example, refuse to insure against Y2k bugs. The Halifax's own contents policy ran from 1999 to 1900, until they noticed that they themselves were victim of the Y2k bug. But if they do not cover people against Y2k bugs, then the manufacturers are not going to be sued by insurance companies. Instead users have become responsible for fixing what — in the best tradition of oppression — they take to be their own personal problem.

Why is this behaviour: the ownership of problems by users, the denial of cover by insurers, the denial of responsibility by manufacturers, and the selling of training and upgrades to fix problems, confined to computer systems? Norman (1998) argues it is because the manufacturers have a 'teenage' mentality; and by keeping products before the threshold, they can stay juvenile as long as they wish.

3.9. HCI as user-centred

HCI problems are apparent by their effect on users, and the purpose of good HCI is to benefit the user. It seems self-evident that HCI should be user-centred. This is the central message of much work in HCI, notably Landauer (1995).

Unsurprisingly, user-centredness is a forceful HCI slogan, but used uncritically it may impede HCI — this paper has given a list of reasons why users may not make sensible HCI decisions! For example, Nader exposed the misuse of "driver error" covering up responsibility for car accidents. Though some accidents are caused by drivers, not all of them are. Analogously, focusing on the user is important for HCI, but it is not the whole answer. Concentrating on user-oriented evaluation might lead, for example, to ways of camouflaging problems rather than avoiding them.

The Y2k problem is an example of poor quality in system programming. Arguably, many systems have poor programming behind them; thus the design quality of products is low and out of control. Therefore to improve HCI one has to concentrate on users and post-design problems. This is a practical response to the problem; but a strategic response — which should also come from HCI — would be to try to improve programming and design standards. For example, there could be usability certification of products, there could be certification of programmers, and so forth.

Christensen (1997) is distinctly *not* user-centred. His argument is that users understand what they are doing not what they might be doing. Users work inside a value network, which may not be the best one for them as technology develops. We should also emphasise designer-centred design (Thimbleby, 1998b) since in fact it is designers who make new artefacts.

4. CULTURE AFFECTS EXPERTS

Professionals working in HCI itself are not immune to the unusability culture. It would be invidious to give examples of particular experts making mistakes (though I have my fair share!), rather Norman (1998) gives a very good summary of companies — such as

Apple, Kodak, IBM — making major errors of judgement.

A typically British story, of being first yet ulitmately failing, is retold in a book poignantly called *User-Driven Innovation* (Caminer, Aris, Hermon & Land, 1996).

4.1. HCI as a natural science

Studying human computer interaction is itself a human activity, and therefore the very limitations that make HCI interesting are exactly the same limitations that make it a hard subject to advance!

Given, then, that HCI is too complex to study it is sensible not to take it *en masse*, but to take manageable parts. For example, we could take the computers as fixed, and study the human issues. Or we could take the human issues as fixed, and just study the computers. Psychology takes the former approach, computer science takes the second.

The conventional approach is for HCI to be treated much as a natural discipline. That is, the world is given, and HCI's job is to find out more about it. In turn this knowledge may help make future systems better. Compare this approach with psychology, which — apart from ethically dubious experiments — has no control over people and takes them as given. Or compare it with computer science, which as an applied area of mathematics, has a sort-of approach that assumes there is an 'ideal' computational model that in principle exists, as yet to be discovered.

To varying degrees, then, conventional HCI can be understood as a science. There is some truth "out there" (whether in people, society, or in a Platonic mathematics) waiting to be discovered. The methods of HCI generally reflect this underlying philosophical approach, that it is or aspires to be a natural science.

HCI lags behind commercial product development, which certainly raises new and interesting HCI issues. Product development is difficult and is the province of commercial organisations, usually requiring significant investment and resources. These facts in turn become an assumption: that commercial products are given (much like the natural world is given) and are therefore proper raw-material for much HCI research. Consider all the research on commercial word processors — compared with the paucity of research developing variations on word processors. Where is the experimental HCI that tries new designs, rather than tries new problems with existing designs?

So whilst many commercial products make interesting guinea pigs for HCI, there is less point in studying failures than in building better systems: HCI is not just a natural science …

4.2. HCI as an artificial science

In contrast to being viewed as a natural science HCI can also be understood as an *artificial* science (Simon, 1970). Nothing is given, because it is created by humans. Seen like this, HCI includes a design element, which creates new systems. Thus in contrast to the usual emphases of HCI, this view emphasises we construct new "truths," by building systems and by training users and requirements engineers … by changing the world.

This view of HCI is more proactive, but even so it is not without its problems. Constructive HCI is assumed to be either computing science or industrial design, and therefore is outside mainstream HCI. Computer scientists have enough trouble getting things to work, without worrying about HCI; and industrial designers make mock-ups that do not need to work before they are commissioned. Thus, whether HCI is a natural science or an artificial science, we still have bad user interfaces!

Thimbleby (1990b) has an entire chapter on the relation of science to HCI.

4.3. HCI lacks boundaries

At a more abstract level, HCI is not a mature discipline, with agreed boundaries.

Many sciences use computers for their own purposes. For example, computational chemistry is a discipline in its own right. Any such science inevitably involves human-computer interaction, and may indeed make specific contributions to HCI itself — for instance, through the discovery or analysis of new effects. Now, when the science is one that is anyway a 'component' discipline of HCI, such as psychology, it is easy to confuse doing it with doing HCI. Is a study of human perception HCI or psychology? Both? (The same might be said of work in computing that involves user interfaces — how can it escape them? — and therefore "is" HCI.) Of course, it does not matter much what it is, except that the confusion leads to the component discipline of HCI increasing its stake on HCI, and therefore seeing alternative views as competitive.

Or since computing science gets "user interfaces to work" it might appear from the computing blur that no HCI is necessary to get interfaces to work — it is all computing. Actually, computing gets

interfaces to work, but not to work well; it is necessary but not sufficient.

4.4. And the lack of theory …

We've described a wide range of reasons why HCI does not get any better. Another reason is that is rather hard to do better without adequate theories.

When there are no theories, problems can be dismissed as coincidences, curiosities, or as "that's just how it is" (Lightman & Gingerich, 1992). Continental drift had been proposed as early as 1800 to explain the close fit of Africa's and South America's shapes, but the idea was dismissed — despite Wegener's data — until the theory of plate tectonics was developed in the 1960s. Both evidence and theory, a demonstrable explanation of the evidence, are required for the problems to be taken seriously.

Clearly HCI needs to develop theories and systems for the reliable design of complex technology, to clarify problems and make them visible at early points in the design process, where they can be managed, analysed and avoided. Unfortunately, HCI is very complex. A complete definition of the user interface of something as simple as a vending machine is at the limits of researchers. Researchers tend to concentrate on either idealised systems, or they gloss the difficulties in knowing what systems are. And without theories, product designers make arbitrary decisions, which in turn are hard to theorise!

HCI as a field is at a pre- or quasi-theoretic stage. Lots of 'small' theories and ideas are available, but industrial practice lags behind because current research does not scale up, nor is it packaged in a way that relates to the needs of designers. Research has polarised into a 'creative' end, proposing new techniques and applications, and an 'evaluation' end that assesses the impact or success of given systems. Proper evaluation is hard to do: commercial systems are poorly specified and what is evaluated cannot be precisely defined. Evaluation and other "outside-in approaches" rarely provide constructive insights that reliably can be fed into new designs, and in any case, designers feel successful enough not to need them — hence perpetuating inferior design.

In the 1980s, computer scientists were often lead designers, but as computers became more accessible and more pervasive, changes occurred: programming, as opposed to graphic designing, became much harder (the programmer-year effort to compete with commercial products increased dramatically); and evaluative approaches (legitimately) took a dominant position in the field of HCI. The result of this has been a down-play in the contribution of theory to design.

4.5. Everyone is an expert

It is very hard if not impossible to work within HCI without having a personal stake in the field. Some of us use Macintoshes, some of us use Unix, some PCs. Whichever we use, we live in a community of users of the same sort of system — many things we do reinforce the wisdom of being in this community. Other people in the same community are more helpful (of course!) and they have solutions we can covet to help us, whereas people working with other systems do not know the answers to our personal problems, and they are not using solutions that we would covet. There are even lively magazines and other fora that emphasise the value of each system to its own community — and often in contrast to the comparative disadvantage of the alternatives.

Almost certainly, then, anyone who finds a way of improving one of these systems is not going to impress the rest of the HCI community!

Likewise, and on a wider scale, we all have video recorders, mobile phones, calculators, and have everyday experience of various sorts of gadgets. Often, then, we have a stake in a particular solution. Other sorts of solution are of no personal interest, because they do not help us use the particular gadgets we own.

4.6. Solutions are hard to publish

HCI as a conventional academic discipline progresses by shared use of refereed literature. Quality work is published, and what is published tends to define the current trends in HCI.

Unfortunately, one of the damaging consequences of the fast-paced development of technology is that for any focused development in HCI, there is almost certainly a commercial product that excels some aspect of that development. Now, since any referee knowing this will have a stake in that technology, they are likely to deprecate the contribution. Thus there is a tendency for the HCI literature to concentrate on procedures and users, rather than new technical solutions. In other words, the literature does not look at alternatives to commercial products; indeed, explicit discussion of products is deprecated as commercial "product reviews."

In short, academic HCI has little relevance to product introduction, because the publishing process tends to avoid such issues — for the reasons given above, and for the more explicit reasons of

commercial reality: academics neither want to libel nor advertise particular manufacturers as it is not professional to do so.

5. WHAT CAN BE DONE?

Manufacturers will not improve their products when their products are consumed by an uncritical public, and the public will remain uncritical while there are no standards on which to base effective criticism.

HCI should include an explicit public understanding of science element. This is valuable in its own right, but is also because HCI should have a vision about changing culture — inspiring, particularly, children to go on to identify and fix usability problems. A spin off would be a greater appreciation of the way in which our society depends on deeper concepts from computing science: it doesn't just depend on the boxes called computers, but depends on programs (and documentation) being well-designed and appropriate for their tasks.

HCI needs more books like Don Norman's classic *The Psychology of Everyday Things* (1988) — renamed *The Design of Everyday Things* for the paperback), and Bell, Witten and Fellows' *Computer Science Unplugged* (1999); however most books, to date, have been "hindsight" or social criticism. HCI needs more constructive approaches: popular books emphasising the deep relevance of HCI design to everyday life, to help consumers be more discerning, and to help designers be more likely to succeed with *new* products.

5.1. A practical way forward

One way of demonstrating the effectiveness and value of research in HCI would be to build a demonstrator system, a sort-of laboratory bench. We will call such a system an Interactive Design Assistant, or IDA. This idea fits 'conventional' HCI; it is recommended by Gould, Boies and Lewis (1991) as one of their four design principles — to integrate design, so that all aspects of usability (user interface, help systems, training, documentation) should evolve concurrently.

An IDA would not just be a simulator, but would have other features. Given a common representation (such as graphics and program), all the following are possible, and can be achieved in an integrated way, consistent with the 'central' program specifying the user interface:

- Simulate user interfaces.
- Provide various usability ratings.
- Generate user manuals.
- Analyse theoretical performance.
- Log and evaluate actual use.
- Animate demonstrations.
- Generate hardware designs.
- … and even arbitrate in usability competitions!

Relating to the public understanding of science theme: an IDA would be able to demonstrate some of the fundamental ideas, such as the wide application of HCI in everyday life, and would make an ideal web site or CD to be associated with such a book!

An IDA would operate on the World Wide Web (using a browser-like interface) and hence make HCI research very widely accessible. It will enable widespread uptake and severe testing of the ideas, as well as far more effective collaboration and support of related research. Other researchers will be able to contribute designs and make them available for critique or as exemplars of particular ideas. An Assistant will automate, and hence make reliable and efficient, many aspects of design, such as quality control and evaluation, and will provide links between all parts of the design process. An IDA would lend itself powerfully to use on the web, such as annotating interface bugs, or presenting usage statistics in novel ways in documentation.

Most designers adopt an appearance-oriented approach, and there are many powerful tools to do this that are very good at visual simulation, but which are (and unfortunately have to be) programmed in an *ad hoc* fashion (Narayanan & Hegarty, 1998; Sharp, 1998). So an important aspect is to package the theory into an easy-to-use IDA to make theoretically-sound methods available and acceptable to the design culture; conversely, the ability to undertake real, large-scale and complete design projects, with evaluation, would challenge the current theories and, in turn, lead to many new insights and developments.

As Shneiderman says (1998), "the advantages of specialised user-interface software tools for designers and software engineers are large. They include an order-of-magnitude increase in productivity, shorter development schedules, support for expert reviews and usability testing, ease in making changes and ensuring consistency, better management control, and reduced training necessary for designers." These are attainable goals for an IDA.

While user interface design remains atheoretic (from a computing perspective), there will be no pressure on anyone to improve. HCI research is based on the claim that considerable improvements are possible, but that the improvements have to be well-founded, integrated, and themselves easy to use. Once packaged and widely accessible, then the evaluation of systems will become much easier. One

might anticipate consumers testing and comparing products with systems such as those proposed, and hence putting greater pressure on manufacturers to design good products, rather than — as they do at present — merely attractive ones, whose usability problems are disguised from purchasers at the time of sale. An IDA, as proposed, would represent a qualitative leverage in design approaches. An IDA would ultimately not depend on any one research individual, group or organisation, and the scope for wider, autonomous, collaboration would be greatly extended.

An IDA would readily support thorough system reviews, somewhat as Dijkstra longed for in his 1972 Turing Award lecture (Dijkstra, 1987). It would also support system design using intellectually manageable approaches, another of Dijkstra's core ideas. As Dijkstra points out, such constructive approaches increase confidence in designs; in contrast, empirical testing, on which so much HCI currently depends (because there is nothing much better) can only show the presence of bugs — and it has driven HCI into a user-centred problem-oriented discipline (see Landauer (1995) for arguments to this effect) to the loss of designers, who could use an IDA.

5.2. Beneficial impact

Why would an IDA help improve HCI?

To be adopted an IDA would have to improve business processes; it would have to speed the time from design to product, or it would have to speed the design iteration cycle (supposing iterative design is being used). Whether an IDA would speed the design process is a matter for its own design to determine — is it fit for purpose? The scheme proposed here is that the IDA would have an open-architecture and run on the Web. It would therefore become a community project, rather like Linux perhaps, and by harnessing such co-operation could likely achieve the productivity gains required.

An IDA would have to improve the market of (good) interactive systems. Businesses need to stay in the market to reap the benefit of better HCI, and this presupposes they are selling products. Here an IDA could help by creating a 'shop front' where users could try products on the web. It would be possible for users (or consumer groups representing users) to program benchmark task sequences: thus a user could select a suite of tasks, and see them run on a variety of designs. Users would then be able to select systems that best suited their needs.

Finally, an IDA would have to reduce after-sales costs. By improving usability, by improving manuals, and by helping provide better task-fit, an IDA should help reduce the numbers of users who buy the wrong products, or who buy products they do not fully understand. Thus, an IDA would show its success in the long term, as companies reduce the level of after-sales support. In turn, this would allow companies to invest in better current designs.

6. CONCLUSIONS

This paper has made three sorts of claim, and each sort of claim can and should be tested:

1. *A range of "cultural" claims.* Are these claims correct? Where is the evidence? What are the confounding factors, for example which would enable the cycle of bad usability to be broken?

2. *A proposal that public understanding of science will help.* Despite its popularity, usability seems not to be treated in the public arena except by journalism, and then usually to celebrate new technologies and new features. There is a considerable scope to exploit the public's interest in consumer gadgets to promote usability.

3. *A specific proposal for a design tool.* Such a tool remains to be built! A companion paper, also submitted to this conference, describes such a tool in more detail.

Acknowledgements

Ann Blandford, Gary Marsden and Prue Thimbleby made many helpful suggestions.

REFERENCES

T. Bell, I. H. Witten & M. Fellows, *Computer Science Unplugged*, 1999. See http://unplugged.canterbury.ac.nz/

D. T. Caminer, J. Aris, P. Hermon & F. Land, *User-Driven Innovation* , McGraw-Hill, 1996

C. M. Christensen, *The Innovator's Dilemma: When Technologies Cause Great Firms to Fail*, Harvard Business School Press, 1997.

E. W. Dijkstra, "The Humble Programmer," *ACM Turing Award Lectures*, 17–32, Addison-Wesley, 1987.

W. W. Gibbs, "Taking Computers to Task," *Scientific American*, **277**(1): 64–71, 1997.

J. D. Gould, S J. Boies & C. H. Lewis, "Designing for Usability — Key Principles and what

Designers Think," *Communications of the ACM*, **34**(1): 74–85, 1991.

T. Landauer, *The Trouble with Computers*, MIT Press, 1995.

B. Laurel, *Computers as Theatre*, Addison-Wesley, 1991.

A. Lightman & O. Gingerich, "When Do Anomalies Begin?" *Science*, **255**(5045): 690–695, 1992.

D. MacKenzie, "Computer-related Accidental Death: An Empirical Investigation," *Science and Public Policy*, **21**(4): 233–248, 1994.

N. H. Narayanan & M. Hegarty, "On Designing Comprehensible Interactive Hypermedia Manuals," *International Journal of Human-Computer Studies*, **48**(2): 267–301, 1998.

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

D. A. Norman, *The Invisible Computer*, MIT Press, 1998.

J. A. Paulos, *Innumeracy*, Vintage, 1990.

M. Piatelli-Palmarini, *Inevitable Illusions*, John Wiley & Sons, 1994.

J. Sharp, *Interaction Design for Electronic Products using Virtual Simulations*, PhD thesis, Brunel University, 1998.

B. Shneiderman, *Designing the User Interface*, 3rd. ed., Addison-Wesley, 1998.

H. A. Simon, *The Sciences of the Artificial*, MIT Press, 1970.

E. Tenner, *Why Things Bite Back*, Fourth Estate, 1996.

H. Thimbleby, "You're Right About the Cure: Don't Do That," *Interacting with Computers*, **2**(1): 8–25, 1990a.

H. Thimbleby, *User Interface Design*, ACM Press Frontier Series, Addison-Wesley, 1990b

H. Thimbleby, "The Frustrations of a Pushbutton World," *1993 Encyclopedia Britannica Yearbook of Science and the Future*, 202–219, Encyclopedia Britannica Inc., 1992.

H. Thimbleby, "Computer Literacy and Usability Standards?" *User Needs in Information Technology Standards*, 223–230, C. D. Evans, B. L. Meek & R. S. Walker, eds., Butterworth-Heinemann, 1993.

H. Thimbleby, "Internet, Discourse and Interaction Potential," in L. K. Yong, L. Herman, Y. K. Leung & J. Moyes, eds., *First Asia Pacific Conference on Human Computer Interaction*, 3–18, 1996.

H. Thimbleby, "The Detection and Elimination of Spurious Complexity," *Proceedings of Workshop on User Interfaces for Theorem Provers*, R. C. Backhouse, ed., Report 98–08, 15–22, Eindhoven University of Technology, 1998a.

H. Thimbleby, "Design Aloud: A Designer-Centred Design (DCD) Method," *HCI Letters*, **1**(1): 45–50, 1998b.