Chapter 2. Does HCI Matter?

Table of Contents

Context

This unit presents arguments for why it is vital to include HCI tools, techniques and practices into the design of interactive computer systems. To do this, four viewpoints are taken: business (to show how user-centred design can impact on performance); quality of life (bad HCI can lead to frustration, anxiety and anger); safety-critical (some systems, if badly designed, can kill); and, the legislation/ international standards are relevant. Many interactive computer systems seem to have been developed with little thought of the user: this unit will discuss the possible reasons for this.

Objectives

At the end of this unit you will be able to:

- Present a business case for using user-centred design.
- Discuss the productivity paradox controversy.
- Discuss the ethical, legal and human issues the use or lack of use of user-centred design.
- Understand why HCI research and practices seems to have had little impact on commercial interactive systems developers.

An Introductory Case Study

The London Eye is a large Ferris wheel erected opposite the Palace of Westminster in celebration of the millennium. It is a very successful tourist attraction offering unrivalled views over London. It makes use of all the most advanced technology from its space age white tubular construction to the fully computerised telephone and Internet booking system.

My parents live outside of London and have been asking me for several months to book them tickets on the Eye for a Saturday afternoon. They suggested a list of dates when they could make it to London and I rang up the Eye's booking telephone system.

A kind, helpful computerised voice patiently explained all my booking options, how long a 'flight' on the Eye would take, where the Eye was situated and what precautions and preparations we should make before our visit. I was then asked to type in the date of our intended visit on my telephone keypad. This I did and the kind helpful voice told me which date I had typed in and then regretfully told me that she was unable to take bookings for that date and suggested I type in another date, which I repeatedly did only to be regretfully informed that she was unable to take bookings for any of the dates which I my parents could make it to London. At this point I gave up and rang to tell my parents that the Eye seemed to be sold out indefinitely. I was mildly irritated by the system, it did not need to tell me all about the Eye first if it was ultimately not going to sell me any tickets. I would have preferred to find out at the beginning of the call whether or not I could get tickets and then be told all the useful information about where the Eye was subsequently if I had managed to book tickets.

A few weeks later I happened to be walking on Embankment and walked past the Eye's booking office. Having nothing else particular to do I went in to enquire from a human being whether the Eye was indeed completely booked for months to come. 'Oh no.' explained the kind, helpful assistant after I

had queued for ten minutes, ‘We reserve some tickets for telephone sales and they have sold out, but you can book tickets in person here.’ So I booked tickets and me and my parents had a wonderful Saturday afternoon looking out over the Thames and being able to see as far as Windsor Castle.

The London Eye only got my business because I managed to overcome the obstacles put in my way by the badly designed telephone booking system. The system did not help me, it hindered me. There was no way I could have found out from the telephone system that there were actually tickets available, even though that information must have been available to the automated booking system. If I hadn't happened to be walking along Embankment with nothing special to do the Eye would have missed out on my business. I can only imagine how many sales the Eye is currently losing because of their telephone system.

This is the broad idea of this unit. Someone in the Eye's management team decided to implement an automated telephone booking system to save money; they would not have to pay expensive human telephonists who get hungry, have to use the toilet and join unions. The saving made to the company by the automated telephone system is there and obvious to anyone auditing their books. (The cost of hiring telephonists would be £x, the cost of implementing the automated system is £y, where y is less than x. Hence a saving of £x – y.) The loss to the company of people not making bookings because of the bad design of the booking system is a lot less obvious. So much so that it is unlikely that the Eye's management is even aware of it.

Activity 1

List all the services which you use which have become automated in the past five years. In particular think about services where you used to liaise with a human being, either behind a counter or on a phone line, and where that human being has now been replaced by a machine. Think critically about each of those services. Have they made your life easier? Or is it harder work for you now?

Unit Outline

In the introductory unit you were introduced to the conception of HCI; what HCI is and what it tries to do. In this unit we will look in detail at the why of HCI. Why it is important and why it fails to have the impact one might expect of it. Designing a system that takes into account its users and is easier to use is always going to be more complicated, time consuming and therefore more costly than designing a system that does not.

Designing for usability needs therefore to be justified carefully, and, assuming that it is justified designers need to be motivated to work in a way that ensures usability.

We will look at justifications from four perspectives:

Business

If producing usable systems is a cost then an argument needs to be produced to show that the benefits of a usable system outweigh that cost.

We outline the ‘productivity paradox'; an economic argument that computers and information technology systems do not seem to pay for themselves. The productivity gains in companies that implement IT system do not seem to significant outweigh the cost of implementing the systems.

The productivity paradox is controversial and has exercised computer scientists for over a decade. We look at the argument from different angles; there are arguments that the paradox does not exist at all and it is merely a myopic use of statistics. There are compelling arguments that the paradox does exist and that it is caused by systems being designed without sufficient consideration for the users of those systems.

We then look at arguments for the cost effectiveness of making systems more usable as well as outlining some success stories.

Quality of Life

Whereas the arguments for developing usable systems may be made explicit and quantifiable in a business context, technology impinges on our lives in ways which may be much less tangible. Badly designed systems can lead to anger and frustration for the users and may reinforce ‘technophobic' attitudes. Only by understanding humans better can such detrimental consequences be designed out of a system.

However, many of these issues are very subtle, take effect in the long term and cannot be easily quantified and costed. We look at how designers are (or maybe compelled to be) motivated to produce systems which are not detrimental to users in the absence of a cost argument.

We look at ethical issues in design, where users have rights not to suffer unusable technology and developers have responsibilities to respect those rights, as well as legal issues where usable systems may be required by law.

Safety Critical Systems

Technological systems are now extensively being put in place in situations where the failure of the system can result in severe injury and loss of life. Whereas ‘productivity' is central to cost effective systems, ‘reliability' is central to safety critical systems, where reliability describes and measures the likelihood that a system may fail with undesirable consequences.

Human operators control safety critical systems and are therefore, in effect, as much a part of a safety critical system as the technology. An understanding of human behaviour should lead to a system being designed which takes into account the behaviour of its operators and therefore is less likely to fail owing to ‘operator error'.

We also look at ethical issues of responsibility; the failure of safety critical systems may dramatically effect the lives of people completely external to the system. In many non-safety critical systems direct legal and ethical relationships can be established between developers, users of a system and consumers of the services provided by the system. Safety critical systems may impinge on a greater stage; if you are knocked down by a motorist driving a hire car whose ABS brakes are faulty, who is responsible? The driver for bad driving? The leasing company for hiring a defective car? The manufacturer? The sub-contractor who developed the ABS system for the manufacturer? Or yourself for crossing the road while knowing that there are cars out there with faulty brakes?

Standards

In many engineering fields ‘standards' are proposed which products should aim to satisfy. Products which conform to those standards are held to have attained an objective level of quality and are therefore more saleable. Standards for usability are immature, controversial and not widely used. Many developers claim that their products are ‘user friendly' but do not often explain what that claim means. Standards offer a way of objectively measuring a product for usability, and therefore a motivation for developers to make more usable products.

These four viewpoints on usability are not exclusive; safety critical system typically must meet cost requirements, etc.

Despite all the motivations for developing usable system we shall lay out, there is very little evidence of user centred design being successfully used in the commercial development of systems. We conclude this unit by suggesting some of the reasons for this apparent failure.

Review Question 1

How does an interactive system differ from a non interactive system? Give examples of types of both systems.

Answer to this question can be found at the end of the chapter.

Justifying Information System from a Cost Perspective

To implement a technology based system is expensive and the cost of doing so must be outweighed by some potential benefits. Typically when an industry makes an investment in IT they expect a pay-back in the form of increased ‘productivity'.

A system of production has inputs (labour, raw materials, energy) and outputs (the product). Productivity is a measure of the value added to the output of the system, a productive system being one where the value of the outputs is greater than the cost of the inputs. Productivity is therefore a widely accepted measure of how well a business is performing. Note that getting your employees to work long hours would not increase productivity, because you are simply making your labour inputs more expensive. To increase productivity substantially you need to work more efficiently, not just harder or longer.

One would expect that the introduction of information technology into a system of production would be just such a tool to increase productivity. IT automates many dull repetitive tasks, either doing away with the need for low-skilled labour (and reducing labour costs) or allowing workers to apply themselves to more skilled tasks, thereby increasing output of quality product (assuming that skilled workers produce better quality products than unskilled workers).

However many economic studies in the United States have shown that the expected increase in productivity caused by the implementation of IT systems simply has not materialised. This is the ‘productivity paradox'.

The Productivity Paradox

Until 1973 productivity increased in the United States, particularly during the 1950s where productivity increased quite dramatically. However since 1973, through the 1980s and into the 1990s productivity in the US has stopped increasing at such a rate; it has still increased, but not nearly so dramatically. This period has been precisely the time during which industry has heavily invested in information technology.

Apparently the economic measures are telling us that implementing information technology has been worthless, it has hardly affected industries' ability to improve their productivity at all, indeed in some of the more pessimistic studies it is claimed that productivity has actually decreased with the implementation of information technology.

Cause and effect

Care needs to be taken with this paradox through. Just because productivity flattened out at the same time as implementation of information technology got under way does not imply cause and effect. The concurrence of these two events may simply be coincidence, the flattening in productivity may be caused by many other factors and it may be that the implementation of IT systems has mitigated the effect.

Landauer (1995) traces this argument through, looking at several ‘econometric' models which try to assign cause and effect to the flattening of productivity. The details are very complex but the procedure is as follows: an estimate is made of what productivity is expected to be and then the difference between expected and actual productivity is measured and candidates (such as crime, pollution, etc.) are suggested that might fill this shortfall.

Landauer argues that total US output is 1.5% below expectations since 1973 and that the ‘usual suspects' cannot be made to explain this shortfall. 1.5% roughly equates to $30 billion per year of ‘missing' productivity. IT investments yielded 13.3% less than expected which over an average

investment of $225 billion results in $30 billion. Landauer suggests that this is the missing $30 billion. The argument is compelling, but not a proof that IT investment caused the productivity paradox.

One should always be very careful with such arguments and hold in mind Benjamin Disraeli's famous maxim: ‘There are three kinds of lies: lies, damned lies and statistics.'

An American phenomenon

The productivity paradox seems to be largely an American phenomenon; it reveals itself clearly in the American economic figures, but is less obvious in the figures of other industrialised countries.

America has shown the greatest growth in the industrialised world in the last fifty years, not least because compared to other countries it did not suffer the wholesale industrial and work force devastation caused by the second world war. Europe and Japan's economies were shattered by the war and much of the late 1940s and 1950s were spent trying to recover. Economic figures for Europe and Japan were therefore completely altered to the extent that trying to ‘factor out' the effect of the war in the economic figures (i.e. trying to calculate how Europe and Japan's economies would have fared economically if it were not for the war) is completely impossible.

It is not really possible to compare the economic figures of America with the rest of industrialised world in a substantive way. The productivity paradox is present in other country's economic figures, but not in such a pronounced way as it is for America.

Computerisation versus information technology

We need to be clear about what we mean by ‘information technology'.

‘Computerisation' has greatly improved productivity, where computerisation is the complete replacement of a low skilled job with automated machinery.

An anecdotal example:

I worked as a bank clerk in the 1980s in a major UK bank, having followed in the footsteps of my father who started work as a clerk in the 1950s. When I started work the process of bookkeeping was almost entirely automated, the calculation of credit and debits was performed by a centralised computer system.

Thirty years earlier the bank employed a great many people to simply calculate customers' balances and maintain their account statements. When a debit or credit was received by a bank for a customer's account then the customers' appropriate account statement would be removed from a drawer and a clerk would write the details of this new debit or credit (in the clerk's best handwriting) onto the statement and then the statement would be refiled. A myriad other ledgers and totals would be maintained by hand and accrued at the end of the day to ensure that all the balances were maintained and errors were not made. If errors were found then an extremely arduous process of checking all the calculations made that day needed to be done to find and rectify the errors.

This process was dull, mind-numbing, and required very great precision from the workers; qualities that are not naturally human, but qualities that are very well suited to computers.

In the 1960s and 70s this process was automated with considerable success, so much so that maintaining a bank account became so cheap for banks that they were able to offer free banking to personal customers and the percentage of the population who had bank accounts exploded. Whereas when my father started work he spent most of his time in a back office balancing ledgers, I spent most of my time liaising with customers, discussing financial packages with them, sorting out problems they may have or simply discussing the weather. I was able to add considerably more value to the bank and their customers than my father could have, even though, all in all, we were paid roughly the same. Hence; a large increase in productivity for the bank.

> Interestingly however, subsequent moves by the banks; into automatic teller machines, phone and on-line banking means that human liaison with customers has been taken almost completely out of the system. My wages are paid directly into my account, I pay for goods by debit card, or by withdrawing cash from a hole in the wall machine and I query my balance by liaising with a computerised voice on a phone line. I cannot remember the last time I actually spoke to a bank clerk.
>
> The banks successfully replaced their accounting functionality by computers and this freed up staff to improve the communications with customers. However recent moves have tried to replace these communications by automated systems. Automated systems are not good at communicating with people (not nearly as good as people are anyway) and therefore it could be predicted that more recent moves to automate banking systems will not show anywhere near the productivity gains that moves to automate the bookkeeping functionality did.

This brings us back to the productivity paradox; computers are very good at replacing people when the jobs that people do can be captured mathematically and are dull and repetitious, but when computers augment a job – where IT is used as a tool by users to help do their job – the benefits of doing so are much more ambiguous. The productivity paradox arises when IT systems that require a large degree of input from their user population are implemented. Hence the theory that it is an inappropriately small concern for users in the design of IT systems causes the productivity paradox. We will return to this later.

Refutations of the productivity paradox

The productivity paradox has exercised the economics and computer science communities considerably, and there have been many arguments advanced that the productivity paradox is simply a statistical mirage.

Productivity is a measure of how well an industry is doing may inappropriate for modern information technologies. Even though productivity has not increased, other measures of how well a company is doing have shown considerable gains. IT has caused an improvement in how industry works, but economists may be looking in the wrong places for those signs.

Productivity as a measure of industrial, not IT success?

> Productivity is a measure of industrial success, and it has been argued that it is no more than that – a measure that is useful when a production system has tangible inputs: raw materials, labour and energy and has tangible outputs: services that can be sold with an explicit cost, or widgets that can be put in boxes and shipped to customers. Systems that have a considerable IT component typically generate products and advantages that are much less easy to quantify than numbers of widgets in boxes.

There has been much recent comment on the emergence of a ‘knowledge economy' where the main product of a system is an improvement in knowledge for the system's consumers. Information technology is about storing information, moving that information around and most crucially translating information from one form to another. A successful IT system will gather raw data, process that data into information and deliver that information to consumers in a timely manner and useful format so that the consumers' knowledge is improved. Much play is made of the fluidity of such systems; out of date information may be worse than useless and the time window on when information may be useful is narrowing all the time.

Until recently stock market prices were reported daily in the financial press and this daily update of information was considered appropriate for most investors. Now technology has allowed stock prices to be delivered to our desktop computers or even mobile phones that is accurate to the minute and the daily update provided by newspapers seems archaic.

Technology can offer these apparent benefits, and there are very difficult to quantify; they are likely to be beyond the scope of equations for calculating industrial productivity.

New technology linked with new management procedures?

Table 2.1. The effect of new management structures and IT on productivity. (From Brynjolfsson and Hill, 1998)

	Little Investment in IT	High Investment in IT
Considerable management restructuring	Small increase in productivity	Large increase in productivity
Low management restructuring	No effect on productivity	Decrease in productivity

Furthermore the implementation of IT systems may be need to be linked with new systems of management within companies. Old industrialised companies tended to have very hierarchical structures with major decisions made at the top of the hierarchy and fed down through layers of middle management with less and less decision making responsibility to the unskilled work force who simply do what they are told.

Computerisation has to a large extent removed unskilled workers, and information technology has allowed the skilled work force access to the information necessary in order to make more responsible decisions. Therefore modern company structure tends to be much ‘flatter' with executive decisions dispersed among employees who have much more responsibility for themselves and their actions.

This flatter management structure is well suited to IT systems and there is evidence that if IT systems and flatter management structures are implemented at the same time then there are the sort of productivity gains for companies that should be expected.

If, however, new management structures are implemented without IT systems then productivity remains quite flat, whereas if new IT is implemented without flat management structures then productivity actually decreases. (See above figure)

This effect may be crucial to the productivity paradox; there are productivity gains in implementing IT systems, but those gains may be obscured by old and outdated management practices and structures.

Management structures may be much more difficult to change than implementing IT systems; management is essentially about the placing and responsibility for power within an organisation. Flattening the management structure is about dispersing that power. In order to implement a flattened management structure the executive must take the decision to disperse their own power, in effect to become less powerful in an organisation that they may own, or have considerable investment in. Persuading the executive to do this is notoriously difficult.

So does the productivity paradox exist

The answer is: probably. But it is unlikely to be as profound as an initial reading of the figures suggest, and it is unlikely to be wholly due to badly implemented IT. But the evidence still asserts that IT has not been nearly as profitable as expected, and it is technology that needs to interact with people that seems to be failing.

Landauer's thesis is that computers are failing to add anywhere near the value that we expect of them. His solution is simple and compelling: improve the usability of computer systems.

Why designing for usability is sound economic practice

Given the evidence for the need for improved usability on a ‘macro' economic level we now look at the actual design of interactive systems. What motivates developers, and how that may bring them gains in the short-term, but trouble in the long term.

Deadlines

Software developers are very conscious of deadlines. The software market is very fluid and developers feel they must get products out into the marketplace on time above everything else. A fear of being usurped by competitors who get their product out before you and corner the market is ever felt and real.

However Cooper argues that this determination to get a product out, no matter what the quality, on time distorts priorities. In the long run, he argues, it is better to get a high quality product out behind schedule, than to get a bad product out on time.

Missing deadlines and failure to ship on time are not as dramatic as often held in the computer industry. He cites as an example the hand-held ‘palm top' market. In 1990 the Penpoint computer from GO was released and it was supposed to herald the dawn of hand held computers. It did not and two years later Apple released the Newton to universal apathy. The in 1994 the General Magic ‘Magic Link' brought out a hand held computer that failed to kindle any dramatic interest. Finally in 1996 the PalmPilot was released, was acclaimed, captured the market and still sells well. The point being that even though it was six years too late, it is a well designed and researched product that satisfies its users.

Features

A typical measure of how good a product is, are the number of features that the product offers.

I am using a now out of date version of Microsoft World to prepare most of these notes. By a quick count I can see one hundred and five options from the menu system alone, each of which relates to some feature or other. Let us say that some of the menu options relate to the same feature, so at a guess I should think there are about seventy or so features. Of these features I use at most ten and I should say I do not know how to use half of the other features, and do not even know what the other features are for. More up to date versions of Word come bundled with even more features, none of which I would use.

I have, in fact, been using Microsoft Word since 1990, and the only new feature that has in any way changed the way I do my word processing is real time spell checking. All other features, buried away in menus, toolbar buttons and other dialogues, stay there, unused.

Adding features does not improve usability unless firstly those features are useful, and secondly those feature are usable.

Furthermore an explosion of features in a system is not neutral. You may think that throwing extra features into a system at worst does not effect usability and may improve it, by offering the user extra things to do. However extra features clutter an interface and may make it more difficult to find useful features. Also the product becomes more intimidating to novice users.

Features can be expensive to develop, but if they are not used then that effort is wasted in the long run.

Throwing mud against a wall

The fluidity of the software market means that there is little emphasis on analysing why a product fails. It is better just to chalk a failure to bad fortune and move on to another venture. The availability of venture capital means that it is easy to do this: venture capitalist will put a little money into a wide range of investments and hope that one at least will be successful enough to make a profit over all the other failures.

A little analysis of why a product fails, and it is typically because a product is simply not a useful tool, or if it is useful it is too difficult to use, will prevent failures in future. A lot of venture capitalists believe that there is no way to predict what will or will not fail (the ‘unpredictable market') and are therefore just as likely to fund bad products as well researched and target ones.

A random activity is light-heartedly called ‘throwing mud against a wall to see what sticks'. The funding of many high tech capital ventures follows very much this idea.

Releasing versions

Software products tend to go through several phases while out in the market place. Successful products iterate through many releases, typically as more and more features are piled in, but sometimes as genuine improvements are made too.

Cooper argues that the real cost in the information age, is not what you are building, but what you are not. If it takes four releases to get your product right then this means you did not release three good products on the first release.

The solution

The solution to all the problems we have outlined above is design, where design is an activity that takes place before anyone actually gets around to writing code. It is important to find and understand the problems that users have and then to design a system that will overcome those problems. It is then important to design that system to be easy enough to use such that the effort of using it does not outweigh the problem it was trying to solve in the first place.

The reason that this does not happen is because of the invisibility of the negative consequences. Recall the London Eye booking system. Its benefit to its owners are a reduction in staffing costs which is easily quantifiable. Its cost is lost revenue, which is invisible and difficult to quantify. Hence a system that is not built with what users actually want in mind. Deadlines and feature lists are all tangible and quantifiable. Management can point with pride to a product that ships on time and has a certain number of features and success has been measured this way because it is easy to do so. Measuring the success of a product in terms of how many useful and usable features it offers is much less easy.

Long term thinking is the key. Rushing out a badly designed product to meet an apparent need may have readily apparent short term benefits. Spending time researching a product requires long term thinking and financing and this may be problematic to justify, particularly if your competitors are rushing out products. Sometimes when you throw mud against a wall some of it sticks, but you are much more likely to get some to stick if you investigate the wall first and design your mud to stick to it.

Summary

Computers have had a great impact on the way we live our lives and do our work. But they have apparently had little impact on how productive we are at doing out jobs. The jury is out as to how little impact they have had, but the evidence of their lack of impact is compelling as well as the argument that improving their usefulness and usability will improve their productivity.

The fluidity and rapidly moving software market means that pressures are placed on developers to rapidly produce badly designed software instead of slowly produced well designed software. The unquantifiable nature of usability mean that there is little effort to produce more usable software and there are also a few myths amongst developers (for example the feature list size = usability myth) which mean that a push for genuine usability is rare.

Review Question 2

What is the ‘productivity paradox'? Explain what may cause the productivity paradox and give an argument as to why the productivity paradox may not exist.

Answer to this question can be found at the end of the chapter.

Review Question 3

Explain why designing systems well should give productivity benefits, but only in the long run.

Answer to this question can be found at the end of the chapter.

Activity 2

In this unit and the next we are going to analyse web surfing from a user centred perspective: that is to say we are going to look at what users actually want to do when they surf the web. We will step through a simple analysis process which requires no special skills other than critical thinking. In total the analysis should take at most three or four hours. We hope to show that there are some usability issues with the most commonly used web browsers.

We hope to dispel some myths about user centred design – that it is woolly, vague and time consuming. We also aim to get you to look and think more critically about software, hopefully moving you from the viewpoint of what the software allows the user to do to the viewpoint of what the user wants to do, and to make you think about why there should be a mismatch between the two.

Complete each step completely before moving onto the next. Each subsequent step builds on the next so try to take notes about what you do in each step; you may need them subsequently.

Clear your desk and switch off your computer, or at least switch off the monitor. You are going to think abstractly about the web and web surfing and you do not want to be distracted by the day to day business of pushing the ‘Back' button or clicking on highlighted links.

Now ask yourself this question: ‘What is the web?' and ‘What does it mean to me?' Think about this carefully and then write down your answer, trying to be as careful and explicit with your description as possible.

Do not use any of the phrases you may have come across in the media like ‘the encyclopedia of tomorrow' or ‘a vast collection of interrelated pages spread out over the world'. If you find the web to be wonderful and exciting then say that, but explain why. Similarly if you find the web ugly and intimidating then say so, but explain why.

Write two or three paragraphs, but think about it before you do. We are trying to make you think about the web, what it is and what it means to you and explain that, and that can be quite difficult. Many people have hazy ideas about what the web is and what its for.

Write down your description, then click here to see mine, but do not look at it until you have completed yours

Now you may completely disagree with my viewpoint, and you are welcome to do so. We are not trying to collect objective information about the web, quite the opposite; we want subjective user opinions, but we want them explained rationally and explicitly.

Save your description. We will refer to it on later activities.

The ethics and legality of usability

In the previous section we looked at financial arguments for getting usability right. In this section we look at some less quantifiable arguments for usability. We claim that designers of software should have a responsibility to the users of their products, a responsibility not to cause more annoyance and irritation to users than absolutely necessary.

What we first want to expose is how if we do not think adequately about other people who we effect by our actions then we can impose our values upon them.

First consider this sad example, which is not really about interactive systems, but demonstrates this principle:

The UK train system is notoriously slow and trains frequently run late. I read in a newspaper a report of an elderly, well dressed, respectable woman on a station waiting for a train. The tannoy announced

that the next train would be delayed by twenty minutes. Most of the passengers milling about on the station rolled their eyes and went to find places to sit down and wait for the delayed train. The woman however burst out ‘How can this be? I am never late to anything! I have prided myself, all my life, that I am on time to everything! Now for the first time I'm going to be late, and it's not my fault!'

The journalist who wrote the article went to console the woman and found that this was the first time that she had decided to travel on a train, having started to feel guilty about using her polluting car.

You may feel that the woman was unnecessarily pedantic, and welcome her to the real world, where we all have to stand and roll our eyes on station platforms. But the fact is the woman had a set of values that were important to her, and that she had placed herself in the hands of someone else's system (and paid for the privilege too) and that system had betrayed her set of values. A train being twenty minutes late is such a common occurrence that most of us shrug it off, and it probably did not even register with the train company as one of their late statistics. But to that customer it was a disaster, and who are we, or the train company, to inflict our lax standards on time keeping to other people?

Now consider this example from Cooper:

The sports car manufacturer Porsche is famous for having great respect for its customers and going out of its way to support Porsche drivers, far in excess of the normal supplier/customer relationship.

In developing their new Boxter model they implemented an electronic fuel flow system. The system would detect air in the fuel injection system and immediately shut down the engine and then disable it until the car had been towed to a garage and serviced. Running out of fuel can severely damage a fuel injected engine.

There was a problem with these systems, and a well known and documented one, namely that if the car was low on fuel (not empty) and went around a corner centrifugal force could move the petrol in the tank away from the fuel feed pipes and let air in. Hence the electronic controls immediately cut out the engine and would not let the car be restarted until the car had been towed to a garage and serviced. An embarrassed Porsche suggested that the only remedy was to open the bonnet and disconnect the battery for a while until the electronic system forgot about the air bubble and allowed the car to start again.

That a high quality car like a Porsche should suffer from such a ridiculous problem, and the laughable solution to it, shows us an important fact: the automotive engineers that built the mechanical parts of the car would never allow such a terrible design out on Porsche customers. But the software engineers who were subcontracted to program the injection system had a very different set of standards and respect for customers than the other engineers who worked at Porsche.

The controllers of the train service in the first example inflicted their set of standards (where being late did not really matter) on their customers. In a similar way the software subcontractors inflicted a different set of standards on Porsche than they would normally consider acceptable for their customers.

Users aren't programmers

Programmers like challenges. Programming can be all about solving logical puzzles and meeting sometimes quite considerable mental challenges. In a lot of cases, however, programmers do not seem to realise that there are a considerable number of people out there, buying their products, who do not enjoy their computer being a challenge. Software should not be designed as a challenge.

In a similar way the ‘remedy' to the problem with the Porsche fuel injection system (to open the bonnet and fiddle about) shows that the designers are expecting the users to behave like they do. Designers of cars like opening the bonnet and fiddling about inside, just as programmers like ‘getting inside' operating system and tinkering around. Drivers and users do not, and it is unreasonable to expect them to.

Most users are not programmers, but the converse is not true. Programmers are users, and therefore may come to the mistaken belief that what they find acceptable and pleasurable to use will be okay for any user.

Landauer gives a collection of statistics about what sort of people find computers easy to use; they are young, highly educated, logical thinking and middle class. In fact, exactly the sort of people who become programmers. Landauer's statistics suggest that programmers design software for themselves.

Both Cooper and Landauer place the blame for useless and unusable software firmly with the programmers who develop it. But they are at pains to point out that this is not because programmers are stupid and unthinking, but precisely the opposite. It is difficult for a programmer to ‘think their way into' a user who does not understand (and does not want to understand) something like a hierarchical file structure, which is completely second nature to a programmer.

What both Cooper and Landauer suggest are ways of structuring, amending and augmenting what programmers do, so that users are better considered.

An unwritten contract between developers and users

As with any other design process there should be an unwritten contract between developers and users. Note that this is different than between the more explicit and legally binding contract between developer and customer. Users need not be customers, in many cases software is bought for users by the companies from whom they work. This puts an extra player in the loop.

Usually a customer buys a product and uses it. If they do not like it they exert economic pressure on the developers by not buying any more of their products or demanding their money back. In commercial settings this loop is extended. Purchasing departments buy software and give (or lease) it to their employees. Those users do not have a direct channel of feedback to the developers and therefore usability may not become a crucial criteria for developers. In any case purchasing departments may not like having their choice of product criticised. It is probable that for every one person who complains to a purchasing department about the products they have bought there will a technical whizz kid who can make perfectly good use of the product, and so what is the complainant's problem? Unfortunately anecdotal evidence shows that for every complainer and every whizz kid there are fifty or so under-their-breath grumblers who get on with using the system, without explicitly complaining, but without explicitly enjoying it either.

The free market is amoral and in this unit we are outlining reasons why software development may push the amorality of the free market into immorality. What is needed therefore is special emphasis by developers, because they in effect control the market, to push their products back into the bounds of ethical and reasonable treatment of users.

Other professions have strict codes and guidelines determining their relationship with and treatment of the public. Software developers lack such ethical codes, mostly because of the immaturity of the field, but as we are discovering their are several subtle barriers to moving software development towards such codes, as well as an ignorance that there is a problem in the first place.

Activity 3

Read the following code of ethical practice the American Society of Civil Engineers [https://www.asce.org/inside/codeofethics.cfm].

Fundamental Principles - Engineers uphold and advance the integrity, honour and dignity of the engineering profession by:

1. using their knowledge and skill for the enhancement of human welfare;

2. being honest and impartial and serving with fidelity the public, their employers and clients;

3. striving to increase the competence and prestige of the engineering profession; and

4. supporting the professional and technical societies of their disciplines.

Fundamental Canons

1. Engineers shall hold paramount the safety, health and welfare of the public in the performance of their professional duties.

2. Engineers shall perform services only in areas of their competence.

3. Engineers shall issue public statements only in an objective and truthful manner.

4. Engineers shall act in professional matters for each employer or client as faithful agents or trustees, and shall avoid conflicts of interest.

5. Engineers shall build their professional reputation on the merit of their services and shall not compete unfairly with others.

6. Engineers shall act in such a manner as to uphold and enhance the hon-or, integrity, and dignity of the engineering profession.

7. Engineers shall continue their professional development throughout their careers, and shall provide opportunities for the professional development of those engineers under their supervision.

Suggest in around 200 words an ethical code of practice for software developers.

Legal requirements for usability

Many countries stipulate health and safety requirements for the workplace. Such requirements have been developed from the study of ‘ergonomics' which looks at physical aspects of design. Offices must contain chairs that can be adjusted to suit the sitter and computer monitors must be adjustable to suit the viewer. Several other physical measurements such as desk elevation, and articles such as foot rests for people whose feet may not reach the floor are legally required.

If ergonomics were begun to be studied during the second world war, then it took about 30 years for their conclusions to filter into legal requirements. By the same measure HCI requirements should now be beginning to filter into law too. Indeed EC Directive 90/270/EEC requires that employers when designing, selecting, commissioning or modifying software should ensure that:

- it is suitable for the task
- it is easy to use, and is adaptable,
- it provides feedback,
- it displays information in format and at a pace suitable for the user,
- it conforms to the ‘principles of software ergonomics'.

This directive has been incorporated into law for member countries in the European Community, although in its current form it is unlikely that anyone would actually be able to successfully bring an action against an employer for transgressing the directive.

In the next section we shall look at safety critical systems, the failure of which can be very dramatic. When such systems fail peoples health and welfare can be seriously affected and therefore they can have much more of a solid basis for recourse to law.

It is interesting to note the difference between the extremes of the American approach to litigation, which seems to be to litigate as much and as often as possible, and the British approach which is to avoid recourse to law as much as possible. We will later look at the Paddington rail crash of 5th October 1999. Subsequent to the rail crash it was announced that no legal proceedings would be taken against any of the parties that may be believed to be responsible for the crash. This caused resigned grumbles in the British press, but utter astonishment to Americans (several of whom were involved in the crash). The threat of legal proceedings mean that people may not wish to give full evidence to the Public Enquiry into the crash for fear of perjuring themselves. The British priority is (nominally)

to find out why accidents occur and put in place procedures to prevent them happening again. The American priority is find out why accidents happen and punish those responsible.

The British approach seems, in the abstract, to be more sensible and level headed, but in reality does not actually seem to work. The Paddington train crash was preceded by the Southall crash for which there was no legal action, a public enquiry and recommendations, very few of which were actually taken up. If the recommendations of the Southall enquiry had been in place, the Paddington train crash would probably not have happened.

Activity 4

Having gathered information about what the web is and what it is for in activity 2 we will now look in more detail at what people actually do on the web. Still without switching on your computer try to think of some generalised tasks that you do on the web. The web is a big store of information, and ultimately you want to read, listen to, or download that information. The problem with the web, which takes up most people's time and effort, is finding your way to that information. See if you can identify strategies that you use in order to get place on the web. Again you may find this difficult to do without actually using a web browser, many users cannot explicitly explain what they do, typically giving utterance to phrases like ‘well I just sort of, well, do it'. This is not good enough to inform a design process, we need explicit descriptions. So, think about what you do and try to write it down carefully.

Make an attempt at doing this yourself before reading mine which is at the end of the chapter.

Again you may behave completely differently to me. So much the better if you do. Save your notes for later.

Safety critical systems

A safety critical system is one which can have catastrophic consequences if it fails. So far we have been talking about productivity; the amount of a value a system adds. In the world of safety critical systems ‘reliability' is crucial. Reliability is a measure of how often and how dramatically a system fails.

In an ideal world we would want no accidents to occur at all, but in the real world accidents will happen, so safety critical system engineers put in an awful lot of effort to ensure the safety of their systems. Essentially they attempt to trade off the severity of a possible accident against the likelihood of that accident occurring. A severe accident is acceptable if it is extremely (very extremely) unlikely to happen. On the other hand a likely accident is acceptable if it is not very severe. Safety engineers work to mitigate accidents by a combination of making them less likely and less severe.

However there have been a number of very high profile and fatal accidents lately, where the failure is put down to ‘operator error'. We will present arguments that ‘operator error' is a misnomer; it is ‘system error' and a lot of the time errors involving users are predictable and preventable.

First consider the following two cases:

The Kegworth air disaster

On 8th January 1989, British Midland flight G-OMBE was flying from the UK's Heathrow Airport to Belfast. The aeroplane was a twin-engined Boeing 737 model with a new computerised set of cockpit controls. One of the blades in the left engine detached and the crew began to start emergency action. The plane could be safely flown with just one engine, so the procedure was to shut down the damaged engine and fly on the working engine. The crew informed ground staff and an emergency landing was scheduled at East Midlands Airport near Kegworth.

The crew were aware of the fact that one of the engines was damaged not only because of instrument readings in the cockpit but because of a loud juddering. The Captain throttled down and shut off one of the engines and the juddering stopped. He had however shut down the good engine and the juddering had stopped just by an unfortunate fluke of aerodynamics. The aeroplane flew on powered only by

its damaged engine. The instrument panel displayed this fact, but not in a way that any of the crew noticed. On final approach to the runway the damaged engine completely failed and the plane began to fall out of the sky. The crew realised the error and then tried to restart the good engine, but could not in time. The plane crash landed onto the M1 motorway just yards short of the runway at Kegworth. 47 passengers were killed and 74 seriously injured. The captain survived.

In the aftermath initial reports stated that both engines had failed and the tabloid press proclaimed the captain as a hero for managing to get the plane so close to the airport and landing it in such a way that many passengers actually survived. After the truth came out in the crash enquiry the same tabloid press denounced the captain and placed the blamed firmly with him.

The Paddington rail crash

On 5th October 1999 a slow moving commuter train left Paddington station in London heading for Bedwyn in Wiltshire. Just outside Paddington there is a large bank of signals on a gantry over the line. The signal for the particular line that the commuter train was on was at red. It was also notoriously difficult to see as it was partially obscured by overhead electrical cables. The driver of the commuter train passed through the red signal and his train was crossing over several lives when it was struck head on by a high speed intercity train travelling to London from Cheltenham. 31 people, including both drivers were killed in the collision and many were severely injured by the fireball which swept through the front carriages of the intercity train. The accident site closed down Paddington station, one of the busiest in London, for over a week, inconveniencing over a million travellers.

Subsequently it became clear that ‘SPAD' (signal passed at danger) events were quite common, and that systems recommended years earlier that would have prevented such accidents had not been implemented. The signalling systems were claimed to be so unreliable that drivers often knowingly passed through red lights. If they did not then the rail system would have most likely ground to a halt.

What is interesting is the reporting of these two terrible accidents; in the case of the Kegworth crash blame was firmly placed with the pilot and crew and they were vilified. However twelve years later attention was much more drawn to the system that the train driver had to use, and the consensus opinion was that blame should lay with the system, and the system needed modifying.

At the time of writing the Paddington rail crash enquiry is just getting under way. It will be interesting to see if it comes to the ‘operator error' conclusion this time.

Operator error

Safety engineers collect and analyse a huge amount of data about the technology they use to build safety critical systems. The hardware which they use will be highly tested with known and documented failure tolerances. The use the most secure software development techniques in order to ensure that requirements for the system are collected and that those requirements actually reflect what is wanted for the system (requirements gathering is a notoriously tricky business). Then software is rigorously developed to meet those requirements. At all stages in the process thorough testing and validation is employed. The engineers expend prodigious effort in ensuring that they ‘build the right system, and the system right'. Many experts with experience of systems similar to the one being developed are consulted and a considerable amount of time and money is expended in order to get a system that is certifiably and explicitly correct.

All too often, at this point this bullet-proof technology is handed to a hapless operator who does the wrong thing with it and can cause a serious accident. In subsequent enquiries the failure is put down to ‘operator error' which is held to be in some way unavoidable. The failure cannot lie with the technology, particularly after all that effort was put into it to ensure its safety. Operator error is a very convenient let out clause for the system developers. It allows then to shift blame away from their product onto operators.

When questioned it is clear that many developers do not consider users to part of the systems they develop; they consider users to be external to their systems. Therefore when users cause errors, the blame for the error lies comfortably outside their system.

A widening of what a system is

What is needed is a wider conception of what a system is. Consider a common safety critical system; the car. Cars are very dangerous, so much so the law insists that their user's (drivers) take training sessions and pass examinations before being allowed to use them. A car without a driver is not a complete system; unless it is rolled down a hill it will not do anything very dangerous without a driver. Therefore a safety analysis of a car which does not take into account its driver would be a fairly sterile exercise.

But many safety critical systems are put in place which do not take their users into account at all. Not only should users be brought into safety analyses, but also much more detail about the environment that is part of the system needs to be considered.

There are a variety of reasons why users are not generally included in safety analyses. Safety analysts like dealing with numbers. Given a piece of hardware they like to know how many times it is likely to breakdown in the next ten years, and the chances are there will be data available to tell them. They like to the know the cost of that piece of hardware breaking down, and there may also be data telling them this too. They can then multiply the cost of breakdown by the number of times breakdown will occur and produce a quantitative estimate of how much running that piece of hardware for the next ten years will cost.

Such data does not exist for humans, or it is very suspect if it does. Because data for reasoning about human behaviour do not fit easily into the processes developed for reasoning about hardware and software, it tends to be ignored. Also information about human behaviour is not ‘black and white'. Arguments can be made about the behaviour of automated computers that can give yes or no answers, but arguments about human behaviour will be much less deterministic. Because of this, taking account of users in safety analyses is rare.

Because questions about human behaviour cannot generally be given in a yes or no way, this does not mean however that no answers can be given. Psychology has developed many models of human behaviour that accurately describe and predict human performance. In particular there are many valid models of human perception that could have pointed to problems with the display configuration in the cockpit of the airliner that crashed near Kegworth.

The problem is to define design processes that can make use of the models of human behaviour developed by psychologists and sociologists.

Review Question 4

How do safety engineers work to reduce the impact of accidents? Why is it crucial to reduce the likelihood of accidents happening?

Answer to this question can be found at the end of the chapter.

Review Question 5

What differentiates the failure of a normal interactive system from the failure of a safety critical system?

Answer to this question can be found at the end of the chapter.

Responsibility for safety critical systems

In the previous section we discussed responsibility for interactive systems. We suggested that developers should move towards being responsible to users as well as customers. Responsibility for safety critical systems should spread even wider. The failure of a normal interactive system impinges on the user and possibly the user's employer; people who have to a certain extent agreed to use the system. Failure of safety critical systems impinges on a much wider stage and can affect people who have in no way made a decision to be part of the system.

Consider the following hypothetical example:

I step out on a road crossing with a car about 200 yards away heading towards me. The car's ABS brake system fails, the driver tries to avoid me but strikes me. The car is a hire car. The ABS system on the car was developed by a subcontractor to the car manufacturer.

Who is responsible for the accident?

You are invited to discuss these issues in discussion 2.

Standards

Previously we discussed legal requirements for usability. Prior to legislating the requirements stated in law are usually tried and tested as ‘standards'. A standard is a collection of guidelines which suggest features which should be designed in or out of a system. They may also suggest performance criteria against which a system can be tested . Once a system has been shown to meet all the requirements placed down in a set of standards then it can be certified as having reached that set of standards.

Standards are set by a wide collaborative process, where many researchers and developers discuss what the goal of a set of standards is, and which guidelines, if met, would actually mean that the system in question met that goal.

Industry has an especial interest in standards. If their products can be shown to met a certain standard then this is a positive selling point for their products. Also if a standard does progress into law then it is imperative for an industry that their products fulfil that standard, and it is best to develop a product keeping a standard in mind that has the potential to pass into law. If the standard becomes law then the product should require little or no work to make it legal.

Because industry has such an interest in standards then it is common industry practice to try and influence standards as they emerge. Industry practitioners sit on standards panels and will attempt to steer standards towards their own products. While this practice may seem a bit unethical it ensures a lively debate about what should be put in standards and also means that industry has to come up with good, explicit reasons as why they develop their products in the way they do. Rather than trying to make up good justifications for a bad product, it is easier in the long run to develop a good product. Also it is important to ensure that the standards procedure is not entirely lead by ‘big players' in industry and that small developers and businesses also have their say.

‘Non-partisan' people from academia and government will also be part of standards panels in order to ensure that the standards set actually challenge industry to improve their products, not just rubber stamp what they do already. But it is also the job of industry to argue that the guidelines and goals suggested by academics are actually practicable.

The development of standards can sometimes require considerable negotiation. Such negotiation is a good thing, because it opens up current design procedures to scrutiny and sets developers goals which may be more than just to develop a product to make as much money as possible.

For a standard to work, however, it must be taken up by developers. The ultimate aim of a standard is to become recognised by consumers, so that they explicitly ask for products which conform to that standard, and also for there to be lively competition between developers to produce products which meet a given set of standards. Standards fall in to disuse, or never get used in the first place, if they are felt to be irrelevant or impossible to implement.

Usability standards and marketing user friendly products

ISO standard 9241 describes in part specifying systems for usability. It gives the following definitions:

Usability: the effectiveness, efficiency and satisfaction with which specified users achieve specified goals in particular environments.

Effectiveness: The accuracy and completeness with which specified users can achieve specified goals in particular environments.

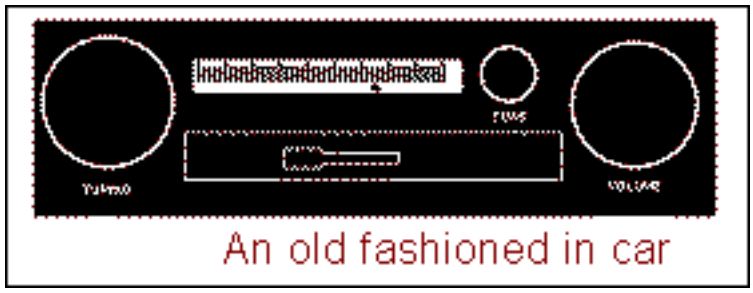
Efficiency: The resources expended in relation to the accuracy and completeness of goals achieved.

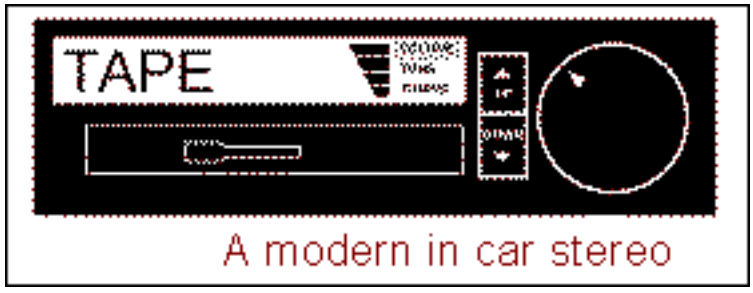
Satisfaction: The comfort and acceptability of the work system to its users and other people affected by the system.

This is a rare example of a usability standard. It describes what ‘usability' is: a combination of factors, effectiveness, efficiency and satisfaction. Each of these are then sub-defined. This process of breaking a definition down into smaller more quantifiable units is popular, but contentious. This reductionist principle means that there is a possibility that an analyst can see and thoroughly measure the trees, but completely miss the wood itself.

Standards for usability are rare, vague and not well developed. The need for such standards is acute however. Many high tech products make claims to be ‘user-friendly', and try to gain a market advantage on the strength of such claims. But in many cases it is not clear where the rationale for such claims comes from. As we discussed earlier implementing ‘user friendly' features is not enough. To be genuinely user friendly there should be arguments and evidence as to why features are user friendly.

Activity 5





The above figures show two controls for in-car stereos. The first is an old fashioned system, with separate dials to control the volume, tone and tuning. The second is a more modern design. It has a menu selection controlled by the two ‘up' and ‘down' buttons and a single dial to set a value. So, to set the volume the user should press ‘up' or ‘down' until Volume is highlighted and then turn the dial to set the appropriate value. A graphic representation of the volume setting is shown on the LCD display.

Draw up two columns, one for each stereo design and write down as many pros and cons for each design as possible. When you have done this weigh up which is the better design from the user's view point. Which one is likely to sell more? Why?

Standards for software and hardware

There are several standards set for hardware both by the BSI (British Standards Institute) and the ISO (International Organisation for Standardisation), but few for software. This is because standards for hardware are based on physiology and ergonomics; fields which are more developed and mature than

psychology and HCI. Hence specific guidelines can be set that will improve the quality of hardware designs.

Because of the less well formed basis of HCI it is more difficult to set specific guidelines and be assured that such guidelines will result in improved usability. Also hardware is less flexible than software and therefore it is easier to set stable guidelines for the design of hardware.

Summary

Standards for usability will emerge as HCI becomes more mature, just as standards for more ergonomic hardware emerged with the maturation of ergonomics. In the mean time experimental guidelines will emerge in the form of draft standards, which may be vague and unspecific, but at least point in the right direction. Standards are important; in their absence developers will be able to ‘get away' with vague and untested claims about their products.

Activity 6

So far we have purposely not touched a computer, instead we have tried to think about and explain web surfing independent of the nuts and bolts of mouse clicks and web browsers. Now we will start to look at actual behaviour.

Given the descriptions of behaviour we produced in activity 4 we now want to generate actual examples of that sort of behaviour. Set yourself a task that will involve you performing in the ways set out in activity 4.

But we need to record how you actually do this. Ideally you should record yourself doing this, if you have a video camera, excellent, if not try a tape recorder and describe out loud what you are doing as you do it. Other ways of recording yourself is to get a friend to watch you and write down what you do as you do it. If all these procedures are impossible then you must try to remember exactly what you did and write it down immediately after you have completed the task. (Question: why do think it is a bad idea to write down what you do as you do it?) It is important that you have a fairly accurate record of how you surfed the web to analyse later. Try to use either Mozilla Firefox or Microsoft Internet Explorer for your tasks. They can be downloaded freely from the Mozilla or Microsoft web-sites.

I described searching and browsing in activity 2, so set yourself two tasks that will get you to behave in those ways. For example in order to get yourself to search for a page, if you have not already done so, search out and read the recommended paper ‘Beyond the productivity paradox' from the ACM digital library web site [http://www.acm.org/dl/]. Alternatively, if you have already done this see if you can find and read Gary Marsden's curriculum vitae from his web pages (he is a member of staff in the University of Cape Town's Computer Science Department. Start off at the University of Cape Town's Computer Science Department [http://www.cs.uct.ac.za].

Now set yourself a browsing task. Try and find a web site dedicated to something or someone you are interested in that you have not looked at before. The web is rife with fan sites for film and pop stars; try to find fan sites for your favourites. Do not stop when you have found one site, most famous people have several sites in their honour, try and find three or four. Again make sure that you are accurately recording how you do this while you do it.

These tasks relate to the behaviours I outlined in activity 4. You should have outlined your own behaviours too. Think up tasks related to your described behaviours and record yourself performing them. Try and be creative, for example I sometimes find that I search and browse simultaneously. A task that conforms to this would be to get several curriculum vitae of UCT staff on screen at once. You will need to launch several browser windows or tabs at once to do this. Also not all members of staff have their curriculum vitae on-line, or indeed have web pages. While you know what you are looking for you will need to browse around the Computer Science web pages looking for what you want.

Set yourself about four tasks and spend a maximum of quarter of an hour on each one, do not worry if you fail to complete a task, in fact failure to complete a task is interesting for analysis. Write out the description of what you did during each task and keep it together with the descriptions from activity 4 that it relates to.

Why usability is not considered an issue

In the first unit we aimed to have persuaded you that there is a problem with interactive systems, namely that they are difficult to use. In this unit we have described arguments that this problem is substantive, that systems that are difficult to use are unproductive, unethical, and maybe even illegal. In the next unit we will start to lay out procedures for designing systems that are easy to use, to alleviate the problem that exists. One question remains in this section: if lack of usability is a problem and procedures exist to improve usability, why are they not already used?

There is not a single answer to this, but a single, simple factor pervades the arguments that follow: software engineering produces products that need to sold on the open market, so if products generally are not usable then there is a lack of demand in the market place for usability. Free market economics dictate that if there is a need that can be fulfilled profitably then someone, somewhere will be fulfilling that need. I could present a detailed, careful argument as to why the word processor on which I am typing this unit is much less than usable, but I, like millions of others, have purchased the word processor as a product over other word processors. I have therefore sent an economic signal to the word processor's developers that I approve of their product. I do not, and neither do any of my colleagues, but we still bought it. Simple economic models do not seem to apply in the purchase of computer software.

We look at several reasons for the failure of the market to produce usable software. None are conclusive, or have a great amount of evidence behind them, but they go some way to explaining the phenomenon.

The dancing bear

Alan Cooper describes a lot of high tech equipment as being like watching a dancing bear. Dancing bears were a popular recreation in the past, a sort of circus attraction. The owner of a trained bear travelled from town to town showing the bear to the inhabitants who would come and stare in fascination at the bear who cavorted and stumbled about in time to music. Everyone was so dazzled by the fact that a bear was dancing, no-one particularly bothered about the fact that the bear did not dance very well.

Technology is analogous; it can perform such amazing feats that we tend not to notice that the way it performs those feats is not very good. On the market there are (quite expensive) 3D modelling package that can produce some quite incredible, beautiful images. However the procedures that the user has to go through to get the package to produce those images are arduous and it is likely that they could be made much easier. But the images it produces are so good users are willing to suspend their displeasure at the interface and interaction problems in order to get to those images. But they have paid money for the system, and by the laws of free market economics, they have sent a signal to the developers of the package that they approve of their package.

It is important to be able to see through what Cooper describes as ‘dancing bearware', to not be so dazzled by the output of interactive system that we do not notice that the process of getting to that output is sub-standard.

Jigsaw puzzles

Many people enjoy jigsaw puzzles: large pictures chopped up into little pieces which need to be reassembled. If a Martian were to conduct a usability analysis of a jigsaw puzzle then they would most likely conclude that jigsaw puzzles were utterly bizarre. The process (reassembling the pieces) of getting to the product (the complete picture) is ridiculously difficult. If people want big pictures why cut them up in the first place? The thing is, people like challenges; to be offered a task that is difficult, but with some effort is possible to complete. Jigsaw puzzles show this quality, they are difficult, but not impossible and the sense of satisfaction of completing a jigsaw puzzle is considerable.

A lot of computer software shows the same qualities as jigsaw puzzles; it is possible to achieve your goals with certain pieces of software, but it just may be very difficult. There is something in human

nature that quite enjoys this challenge. Video recorders are notoriously difficult to program to record programs in advance, but the sense of satisfaction in getting them programmed and the right TV shows recorded can be considerable.

A lot of computer software shows the same qualities as jigsaw puzzles; it is possible to achieve your goals with certain pieces of software, but it just may be very difficult. There is something in human nature that quite enjoys this challenge. Video recorders are notoriously difficult to program to record programs in advance, but the sense of satisfaction in getting them programmed and the right TV shows recorded can be considerable.

Improving usability would reduce the jigsaw puzzle factor in technology and disenfranchise the office guru.

Programmers aren't software designers just as brick layers aren't architects

Programmers produce software. What is important to programmers is designing algorithms that deal with tricky technological issues. This sort of problem solving is highly skilled and crucial to building software. But what programmers are not necessarily skilled in is designing solutions to users' problems. This is a different, crucial, but neglected skill in system development. Much software shows signs of not being designed to solve the problems of users, but being designed to solve programmers' problems, and what is important to programmers is not necessarily important to users. Users do not care what language their word processor is developed in so long as it works, just as they do not care about the mortar that holds the walls of their house together so long as they do not fall down. This is not to decry the necessity of building word processors in an appropriate language, or the necessity of making decisions about the composition of mortar in walls, but a user and house holder should not need to know or care about them.

We do not expect brick layers to design houses, that is not their expertise, we expect architects to design and brick layers to build. Architects and brick layers have different, complementary and crucial skills.

Possibly because of the immaturity of software engineering as a discipline (it is at most forty years old, architecture is several millennia old) so far the brick layers have run the show, without bringing in the skills of architects. There are several more reasons for why there is inertia to change this way of producing software detailed in Cooper (1995).

Technological arrogance

Technology based system support is strangely arrogant and critical of its users. Many Usenet groups are set up to discuss certain pieces of technology. Many times novice users post questions to such groups only to receive curt responses telling them to go and read the manual. Such an attitude to users is a disgrace, but very prevalent. It belittles users and makes them feel stupid, they are therefore less likely to complain about bad software because they are more likely to attribute failure to their own imagined stupidity than to the software.

There is a maxim that there is no such thing as a bad child, only bad parents. In the same way there should be a maxim that there is no such thing as stupid users, only stupid software. However an arrogant culture is very resistant to such a maxim and will act to ensure that users still feel inadequate and blame themselves for failure, rather than blaming the software developers.

The correct response to told to read the manual to tell the developers to write the software so that users do not have to read the manual.

Cognitive dissonance

There exists an interesting psychological phenomenon known as ‘cognitive dissonance', which pops up in many areas of human behaviour. An interesting aspect of cognitive dissonance theory states that if someone does something illogical once then they are likely to repeat this action. This is because we

do not like to think of ourselves as irrational people, so if we do something irrational we will try to convince ourselves that doing it is actually not all that illogical. Therefore we may repeat the action to keep convincing ourselves of its rationality.

Purchasing computer software shows many of the characteristics of cognitive dissonance. We buy badly designed systems once (because we have little choice not to), realise that we have bought a dud but try to convince ourselves otherwise by buying more bad software. Buying software is a strange process; it can be very expensive, but has little physical presence. If you spend a thousand pounds on books you will get a satisfying large stack of books to put on your bookshelf. If you spend a thousand pounds on software you may get a single CD-ROM or even just a password so that you can download the software off the Internet. This unreality and lack of tangibility in buying software may add to the cognitive dissonance effect.

Review Question 6

Explain why ‘buyer beware' is not an adequate or appropriate maxim for computer software products.

Answer to this question can be found at the end of the chapter.

Activity 7

Now we look in detail at the web browsers you are using to accomplish the tasks you set yourself activity 6. Web browser software has many functions associated with it. Accessing these functions may be a case of pressing the buttons along the top of the browser or selecting options from the menus. Write down each of these functions on a piece of paper, systematically going through each of the menu options and buttons on the browser. Spend a couple of minutes on each function, writing down the name of the function and what it does. Try to explain what each function does carefully and clearly. For example, do not just say that the ‘Save' function saves things to disk, try to explain what it saves, in what format and to where. If you do not understand what a function does investigate it for a couple of minutes and see if you can get a better idea. If not, write down ‘Not understood' next to that function. Do not use the help facility to find out what a function does.

Now let us just stand back for a moment and look at what we have done. We are, if you have not guessed already conducting a user centred analysis on web surfing. Activities 2, 4, 6 and 7 are the data collection steps. We have collected opinions about what the web is, what it is or is not good for, what sort of behaviours users employ when looking around the web and we have made some explicit recordings of those sort of behaviours. We have done all the from the user's point of view. We have not concerned ourselves with technical issues like TCP/IP or operating systems. From the user's point of view these issues should be irrelevant.

In the next unit we are going to analyse the data we have collected and suggest some redesigns to web browsers so that they match better what users actually want to do.

At this stage it is a good idea to reflect on what you have done so far, and think about the browser you have been using. Does it allow you to do everything you wanted to do? Or did you find that it got in the way sometimes? Did you feel that you were interacting with the web or with the browser? This distinction is crucial: a good web browser should not make its presence felt. Think of a television; a usable television allows you to watch programs without particularly worrying about the technology in the set which brings the pictures to you. You should be watching television programs not the televisions themselves. In the same way you should feel that you are interacting with the web, not with the web browser. Have a think about this and make notes about your interactions so far; describe the good interactions where you found what you wanted quickly and easily, and he bad interactions where you felt that you were chasing your tail. Try and explain why you felt this.

Conclusion

You may find that we have been rather pessimistic in this unit. We have been roundly rude about several working systems and we have also been roundly rude about the developers of those systems.

We have questioned the value of implementing IT systems in the first place and described some extremely unpleasant consequences of the failure of safety critical interactive systems.

The conclusions we want you to take from this unit are as follows:

1. Implementing IT is not a panacea, and will not automatically increase the productivity of your business.

2. Programmers are very good at programming and should be celebrated at such. But because someone is good at solving problems in implementing software, it does not mean that they are good at solving human work problems. Cooper's thesis is not that programmers are bad at their jobs, but that they are doing in effect too much; they are designing systems for users when that is not their skill. There should be people skilled in this area as a complementary part of the design team.

3. If users are considered to be part of safety critical systems and not external to them, then the systems can be designed to take better account of user behaviour. Safety critical systems will therefore be less likely to fail owing to ‘operator error'. ‘Operator error' should not exonerate system developers from blame for system failure.

4. Mechanisms (legal, ethical and standardisation) exist in many other areas of engineering to motivate engineers to improve their product. There is no reason why these mechanisms should not apply to interactive systems.

If this unit has been negative in tone, by pointing out problems, subsequent units should be more positive, by suggesting solutions.

Discussion Topics

Discussion 1

Discuss the British and American approaches to litigation. How do they compare to your country's approaches. Which is ‘best'? What do you mean by ‘best'?

Discussion 2

Consider the hypothetical accident in the section ‘Responsibility for safety critical systems'. What do you think? Write down each of the people who might be responsible for the accident and write down at least one reason why that person is responsible and at least one reason why that person is not responsible? Against who can you place the most, or most compelling arguments that they should be held responsible?

You can find some points about this discussion at the end of the chapter.

Answers and Discussions

Answer to Review Question 1

- Business. Implementing systems that are designed to be usable, but running systems with usability problems can be much more of a cost in the long run.

- Ethical and legal. Designers should have a responsibility to the people who use the artifacts they design. This responsibility should be to not cause more work, frustration and upset than absolutely necessary. Many countries have legal systems which aim to uphold the rights of people using systems. In many case these are being extended to users of computer systems.

- Safety critical systems. Many computerised systems are now in place the failure of which can be catastrophic. In many widely reported case the failure of those systems has been put down to

‘operator error'. Understanding operators and how they behave will lead to systems being which are more insulated to ‘operator error' and less likely to suffer from catastrophic failure.

- Standards. There is a move to define a set of objective standards for usability, so that products can be measured and certified as being usable.

Answer to Review Question 2

The productivity paradox is the two apparently surprising facts that: productivity rates in the US have flattened out considerably since 1973, and this is the time that industry has invested in IT.

It would appear that investment in IT has not caused any particular improvement in productivity, indeed it may have reduced productivity.

Landauer argues that IT systems that have to interact with users, or which are intended to support users in their jobs (not completely replace them) are badly designed. They do not take their users into account well enough, hence users have difficulty using them efficiently and this is detrimental to productivity.

It has been argued that productivity is an inadequate means of measuring success in information industries, because information industries generate intangible products that are difficult quantify. Furthermore if new IT investment is coupled with new management structures that make the best of the IT systems then productivity improvements are apparent.

Answer to Review Question 3

A well designed IT system is more likely to

- be actually used,
- be used well and efficiently, and
- give pleasure to its users.

Because of this users and their employers are much more likely to get a payback dues to tasks performed well and efficiently using the IT system. Badly designed systems are cheaper to develop and buy in the short term, but do not give the benefits described above.

Answer to Review Question 4

Safety engineers work to reduce the impact of accidents by:

- reducing the likelihood of the accidents happening, and
- reducing the severity of accidents.

If the severity of a possible accident cannot be significantly reduced then its likelihood must be significantly reduced and vice versa. However, because the consequences of an accident for many safety critical system cannot be accurately predicted, work to reduce to likelihood of accidents is preferable.

Answer to Review Question 5

The failure of a normal interactive system can be irritating and time consuming, but typically causes no lasting damage. If safety critical systems fail the consequences can be severe and permanent.

The failure of normal interactive systems only effects the user and possibly the user's employer. The failure of a safety critical system can affect a great number of players, external to and not complicit in the working of the system.

Answer to Review Question 6

Free market maxims like ‘buyer beware' do not often apply to computer software, because it is often not the case that the buyer is the one that has to beware of the product being bought. Users of bad products usually have considerable leverage over developers by simply refusing to buy their products any more. In the software market, particularly for commercial products, the people who buy products are not the same people as use them, therefore the usual economic channel of feedback is blocked.

Discussion of Activity 2

My viewpoint of the web is as follows: ‘The web is a vast array of information and with some effort the chances are that I'll be able to find some information about any topic I'd care to name. I am, however, suspicious of the accuracy of that information and use the web more as a tool to point to valid information, rather than as the source of that information. I find it enormously frustrating trying to find information on the web and rarely search around the web as whole, instead I have a collection of pages in my bookmarks that point to interesting places and 90% of my time is spent in those pages or one or two links away from them.

'The content of the web is claimed to be global and universal, but I am suspicious of that claim, I find the information there very culturally-centric around the North American way of life and viewpoint. Also the demographic of the web is largely affluent, white and male and the content of the web reflects this. I find many of the places on the web where interaction takes place with others to be unpleasant; chat-rooms, Usenet groups and mailing lists have an unfortunate tendency to be aggressive and pedantic, qualities I find unhealthy. I tend not to hang around long anywhere I encounter that attitude.'

Discussion of Activity 4

‘I have two general strategies for finding things. The first one I call searching, where I know what I'm looking for, and I'll know when I've found it, I just don't know where it is precisely. My strategy in this case is to start off on a page from my bookmarks that looks as though it may be close to the desired page and then follow likely looking links. I'll do this for about ten minutes or so. If I haven't found the page by then, I'll resort to a search engine, type in search terms that are relevant and move on from there. If, after another ten minutes, I still haven't found the page I'll either give up or try and find a colleague to help.

The second strategy I call browsing. Typically I have a vague idea about what I want to find and no idea about where to find it. I'll usually start from a search engine and enter a few terms, read through the results and then alter the search terms if they don't seem to be generating the sort of results I'm after. I'll then start investigating the pages suggested by the search engine. I'll follow a link and then explore that page, following links from there to interesting look pages, or backtracking to the search engine results and then repeating the procedure from other likely looking pages. At times during this process I may notice links to pages that are interesting, but nothing to do with the original topic I was investigating. I may then completely change the goal of my browsing and search after that topic instead, or I may bookmark that page and come back to it later.'

'The content of the web is claimed to be global and universal, but I am suspicious of that claim, I find the information there very culturally-centric around the North American way of life and viewpoint. Also the demographic of the web is largely affluent, white and male and the content of the web reflects this. I find many of the places on the web where interaction takes place with others to be unpleasant; chat-rooms, Usenet groups and mailing lists have an unfortunate tendency to be aggressive and pedantic, qualities I find unhealthy. I tend not to hang around long anywhere I encounter that attitude.'

Some Points for Discussion 2

The following may be responsible:

- The car driver: for bad driving, but is reliant on the car's brakes that failed.

- The hire car company: for leasing out a defective car, but is reliant on the manufacturer to supply a working product.
- The car manufacturer: for supplying a defective product, but is reliant on the subcontractor to produce working ABS.
- The ABS system subcontractors: for building defective ABS, but reliant (presumably) on manufacturer to test their system.
- Myself: for walking into middle of road knowing that there are cars out there with bad brakes.
- No-one?

The point is not that anyone in particular is responsible, but that the consequences of the failure of safety critical systems are widespread. In some cases it cannot actually be predicted who will suffer the consequences of system failure and therefore it is essential to reduce the possibility of the accidents happening in the first place.