Chapter 1. Interactive Systems

Table of Contents

"I think there is a world market for maybe five computers." - Thomas Watson, chairman of IBM, 1943

"There is no reason anyone would want a computer in their home." - Ken Olson, president, chairman and founder of DEC

Context

This unit introduces the specialist computing area known as Human Computer Interaction (or HCI for short). One aim of the unit is to give you a clear understanding of what practitioners and researchers in this field know and do. Importantly, too, the unit will try to motivate you to see that HCI is a vital component of successful systems development by giving success stories where the user was considered in the design and failures where the users were overlooked. The unit will also outline the other units in the course and give full details of the assessment scheme used.

Objectives

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

- Understand the nature of an interactive computer system
- Distinguish between design and evaluation processes in an interactive computer system
- Place the specialist area of HCI within the broader area of Computing and systems development.
- Argue the importance of HCI practices within interactive systems development.
- Know what is required to complete the course – the knowledge, skills and assessments involved.

Interactive Systems

The Past

This module is about the design and evaluation of a class of systems that were not even envisaged until relatively recently.

Until the 1980s almost all commercial computer systems were non-interactive. Computer operators would set-up the machines to read in large volumes of data – say customers bank details and transactions – and the computer would then process each input and generate appropriate output.

The Present

There are still lots of these systems in place but the world is also now full of interactive computer systems. These are systems that involve users in a direct way.

In interactive systems the user and computer exchange information frequently and dynamically. Norman’s evaluation/execution model is a useful way of understanding the nature of interaction:

1. User has a goal (something to achieve)
2. User looks at system and attempts to work out how he would execute a series of tasks to achieve the goal.
3. User carries out some actions (providing input to the system by pressing buttons, touching a screen, speaking words etc)
4. System responds to the actions and presents results to the user. System can use text, graphics, sounds, speech etc.
5. User looks at the results of his action and attempts to evaluate whether or not the goals have been achieved.

A good interactive system is one where:

- User can easily work out how to operate the system in an attempt to achieve his goals.
- User can easily evaluate the results of his action on the system.

Note

This course is about methods, tools and techniques that can be used to ensure that the user and computer can interact effectively. We will be looking, then, at the Human-Computer Interaction (or HCI) elements of systems design.

What interactive systems do you use in your day-to-day life? Your first response might be to identify the Personal Computer and all its applications. However, the term ‘interactive system’ can be applied to a much broader range of devices, for example:

- Mobile telephones
- Cash dispensing machines
- The World Wide Web
- Car navigation systems
- Video recorders
- Machines driven call centres (e.g. for telephone banking).
- Workflow system to co-ordinate a teams work-efforts.

Activity 1 - A diary

Over the next week, keep a log of all the times you use an interactive computer system. For each encounter record:

- The goal you were trying to achieve.
- How easy the interactive system was to use as you attempted to complete your tasks.
- Any problems or frustrations you had with the system.
- Any improvements you would suggest to the system.

A discussion of this activity can be found at the end of the chapter.

The Future

In his book, ‘The Invisible Computer’Don Norman argues the case for ‘information appliances’. He suggests that the PC is too cumbersome and unwieldy a tool. It has too many applications and features to be useful. He sees the future as being one where we use specific ‘appliances’ for specific jobs.

Norman envisions a world full of information appliances, a world populated by interactive computer systems:

- The home medical advisor: sensors in the home will enable blood pressure, temperature, weight, body fluids and so on to be automatically monitored. A computer could use these readings to assist with medical advice or to contact a human doctor.
- Digital picture frames: give this frame to a friend or relative. When you have taken a new picture you want them to share, simply ‘email’ the picture direct to the frame. The frame will be connected to the net wirelessly.
- The weather and traffic display: at the moment, when we want the time we simply look at a clock. Soon, perhaps, when we want to know the weather or traffic conditions we will look at a similar device.
- Embedded systems within our clothes: ‘consider the value of eyeglass appliances. Many of us already wear eye glasses … why not supplant them with more power? Add a small electronic display to the glasses … and we could have all sorts of valuable information with us at all times’ [Norman 99, pg 271-272]

Many people believe we will soon enter an age of ubiquitous computing – we will be as used to interacting with computing systems as we are with other people. This dream will only be fulfilled if the businesses that produce these systems and services clearly understand the needs of users so that the systems can be useful and usable.

Review Questions 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.

Review Questions 2

What does the term ‘ubiquitous computing’ mean?

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

Usefulness and Usability

Back Story

DotDash Bank PLC has launched a new telephone-based banking service. Customers will be able to check balances, order chequebooks and statements and transfer money all at the press of a button. Users are presented with lists of choices and they select an option by pressing the appropriate touch-tone key on their handset. The system development team is certain that the system is technically very good – the speech synthesis used to speak out instructions/ options is the state-of-the-art and the database access times are very fast.

The new banking system described is clearly a success from a system point of view: the designers have thought about the technical demands of the system to achieve, for example, high through-put of database queries.

How, though, do users feel about the system?

Note

The bank’s customers have responded badly to the new system. Firstly, users want to know why the system does not let them allow them to hear details of their most recent transactions, pay bills and do other common functions. Worse still, they find the large number of key-presses needed to find out a piece of information tedious and irritating. Often, users get lost in the list of choices, not sure of where they are in the system and what to do next.

From a human perspective the system is a real failure. It fails because it is not as useful as it might be and has very serious HCI problems – it fails because the designers have not fully considered what would be useful and usable from the customers’ point of view.

Usefulness

For an interactive system to be useful it should be goal centred. When a person uses a computer they will have one or more goals in mind – e.g., ‘work out my expenses for this month’; ‘buy a book on motor mechanics’. A useful interactive system is one that empowers users to achieve their goals. When you build an interactive system you should make sure you use a range of design and evaluation methods to discover the goals and associated system functionality that will make your system useful.

Usability

A cork-screw is a tool for opening bottles sealed with a cork. They are useful tools. However if you are a left-handed person most cork-screws are difficult to use. This is because they are designed for right-handed people. So, for a left-handed person the cork-screw has low usability (despite being useful).

Usability is about building a system that takes account of the users' capabilities and limitations. A system that has good usability is likely to have the following qualities:

- Flexible. Users should be able to interact with a system in ways that best suit their needs. The system should be flexible enough to permit a range of preferences.
- Robust. A system is robust if a user is given the means to achieve their goals, to assess their progress and to recover from any errors made.

In a later unit we will look at each of these aspects and consider ways in which they can be achieved.

Review Question 3

What qualities does a usable system have?

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

Why is HCI important?

Note

‘Interfaces are something we do at the end of software development. We want to make the system look nice for the end user’

Unfortunately many analysts and programmers might agree with the above statement. They cannot see the point in spending time and money on seriously considering and involving the users in design. Instead they consider they know what is best for the user and can build effective interfaces without using extensive user-centred methods.

However experience has shown that badly designed interfaces can lead to serious implications. If you build poor interfaces you might find:

- Your company loses money as its workforce is less productive than it could be
- The quality of life of the users who use your system is reduced
- Disastrous and possibly fatal errors happen in systems that are safety-critical

Productivity

There has been a lot of interest in the past into a phenomenon known as the productivity paradox. People have wondered why when so much money has been spent in recent years on computer systems has there been such a limited improvement in organisation and country productivity. The common belief is that computers can make a business more efficient and effective but there has been little economic data to back this up.

Tom Landauer, in his book The Trouble with Computers, suggests that one of the key reasons why computers are not living up to our expectations is that they are difficult to use. Employees waste time with difficult systems, become frustrated and slowed down and are thwarted from achieving their employer’s goals.

Quality of Life

Note

‘I booked a ticket to fly to Athens for an Easter holiday using an online e-commerce site. I was impressed by the cheapness of the ticket and that I could do all the booking without leaving my computer.

The time for the holiday came and I drove to Gatwick airport in London. I parked my car and went straight to the check-in. Everything went well until the airline representative said, “You do know that you are flying back from Athens into London Heathrow, don’t you?”.

We had a good holiday but when we returned we landed in Heathrow and then had to spend 4 hrs trying to get to Gatwick airport (some distance away).

I felt let down by the Web site I had booked the tickets at.

A poor user interface had led to stress and tension for the holiday maker. A well designed interface might have meant that the problems never arose.

Activity 2 - Redesigning the Online Flight Booking System

Below is an extract that appears in the discussion on usability and quality of life:

Note

‘I booked a ticket to fly to Athens for an Easter holiday using an online e-commerce site. I was impressed by the cheapness of the ticket and that I could do all the booking without leaving my computer.

The time for the holiday came and I drove to Gatwick airport in London. I parked my car and went straight to the check-in. Everything went well until the airline representative said, “You do know that you are flying back from Athens into London Heathrow, don’t you?”.

We had a good holiday but when we returned we landed in Heathrow and then had to spend 4 hrs trying to get to Gatwick airport (some distance away).

I felt let down by the Web site I had booked the tickets at.

How would you redesign the online booking Web site interactions to avoid such problems?

A discussion of this activity can be found at the end of the chapter.

Safety-critical systems and disasters

Sometimes poor HCI design can lead to very serious implications. There are many systems that are safety-critical. A safety-critical system must work under all conditions without failure or error. Some examples are:

- Aeroplane control systems
- Nuclear power control systems
- Computer controlled medical equipment.

A Near Miss

Note

A new type of aeroplane had all its documentation online. Pilots could browse through the material in the cockpit as if they were browsing the Web.

On one particular flight the pilots encountered a potential serious engine problem. The captain tried to use the interactive documentation system to get help. Unfortunately he became lost in the maze of inter-linked documents.

By chance the co-pilot had printed out the appropriate part of the manual the night before (he was revising for an examination) and the captain quickly found what he needed to know by flicking through the notes.

Bad interfaces can lead to disasters and even fatalities.

In unit 2 of this course we will think in more detail about the importance of HCI. You will explore arguments that you could use to convince your company to take interactive system design seriously.

Review Question 4

Imagine you are the technical director of an electronic company that makes a wide range of product including home entertainment systems, safety-critical systems and business productivity tools. You wish to persuade the board of directors that they should employ a number of Human Factors people. Write some outline notes of the presentation you would make to the next board meeting.

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

Designing and Evaluating usefulness and usability

This course is about the design and evaluation of interactive computer systems in order to improve usefulness and usability. What does design and evaluation mean and why are these activities important?

Design

When a child builds a house out of bricks there is very little design. It is unlikely that the child has analysed requirements, drawn up plans (with alternatives) and selected tools and techniques to carry out the task.

Unfortunately, sometimes when an interactive system is built, designers fail to consider an essential aspect of the system – the human users. For successful interactions, there has to be explicit and well-thought out consideration of this human factor – usability needs to be designed into the device or system.

To achieve this, as we will see later in the course, there are a range of models, techniques and tools that can be used to construct the system. All of these methods attempt to centre the design on the user group (this is why the methods are collectively known as User-Centred Design or UCD).

Interactive system designers must understand their users. A design developed for one type of users might not be the best for another group. For example, think about a cash-dispensing machine. What would be a good interactive design for this sort of system? Unless you know something about the user group you will not be able to give a workable answer to this question.

Activity 3 - Design for Diversity

Interactive system designers need to know their users if they are to build effective systems. User-centred design, then, involves really understanding the capabilities, limitations, needs and wishes of your user group.

Standard cash machines (ATMs) are designed for ‘standard’ users with no physical or mental impairments.

Imagine you are designing an ATM for Sue. Sue is 75 years old.

- List out the possible physical and mental characteristics that Sue might have and that are relevant to the design of an ATM.
- How would you design an ATM to better suit Sue?

Now look at the World Wide Web Consortium's advice [http://www.w3c.org] designing Web sites for universal access.

A discussion of this activity can be found at the end of the chapter.

Evaluation

Note

imagine a parachute designer has designed what he feels is the ultimate parachute: lightweight, can carry heavy loads, and very safe. He is so convinced that it will work that he does not test his product until it is has been built. Then, one day, he takes a plane to a high altitude and, putting his faith in the chute, jumps…

A good designer will not simply trust that his skill and experience will always produce designs that are highly effective. Evaluation is about testing to see if the interactive system has good usability and usefulness. There are two types:

- Formative evaluation: it is not good enough just to test your system once it is completely built. Evaluations should be carried out all the way through the design and development cycle. The results of these evaluations should be used to guide the design.
- Summative evaluation: once a system has been built then an overall assessment of its usability is needed. These tests should be done to validate aspects of the design (e.g., is the system as learnable as we specified?) and to test the acceptance of the system by the end-users (i.e., do the users like the system, find it easy to use etc?). If a company is buying an off-the-shelf system, it might carry out summative evaluations of all the products to see which is best.

In this course you will look at several evaluation methods and learn when and how they can be used appropriately.

Activity 4 - Looking at bad designs

Spend half-an-hour looking at examples of bad human factors design on Michael Darnell’s Web site [http://www.baddesigns.com/].

- Before reading why the design is bad, spend some time carrying out your own summative evaluation to spot the problems.
- Choose 3 items examples that you found the most interesting and post your thoughts on the course bulletin board.

A discussion of this activity can be found at the end of the chapter.

Review Question 5

What is the difference between summative and formative evaluation? Why are both needed in the development of interactive computer systems?

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

The HCI Discipline

History

The study of Human-Computer Interaction has developed into a discipline in its own right over the last ten years or so.

Long before HCI people were studying how humans and systems (machines, processes and so on) worked together. In the Second World War, for example, weapons developers were interested in making their products more effective (!).

There has been a lot of work into the ergonomics of the machines, environments and systems that humans are involved in. The Ergonomics Society in the UK recently celebrated its 50th birthday (it was founded in 1949).

Ergonomics (which is sometimes also known as Human Factors) is mainly concerned with making sure that the physical aspects of a system fit well with a human’s capabilities and limitation:

- Car design: are the cars controls – steering wheel, dashboard, pedals etc – well placed for the drivers?
- Product design : is this tool easy to handle?
- Workplace: is the lighting good; is the desk and seat adjustable to suit the users?

As more and more computer systems were introduced, people began to get involved in thinking about the way in which humans relate and interact with these new environments. Initially this new area of work was called man-machine interaction; it is now much more aptly named human-computer interaction.

Who does HCI

Researchers

Many universities and technology companies (e.g. Xerox [http://www.xerox.com], Apple [http://www.apple.com], and Microsoft [http://www.microsoft.com]) have research labs that are dedicated to improving interaction by amongst other things -

- Finding out key aspects that affect the quality of interaction.
- Developing new technologies (e.g. handwriting recognition) for interaction.
- Developing models and tools which system builders can use to build better interfaces.
- Evaluating the impact of alternative interaction approaches on usability.

Practitioners

Organisations that build interactive systems need HCI professionals to help them do a better job. Some examples of HCI in practice are:

- A HCI Lab in a consumer electronic manufacturer. Here, the HCI specialists are involved in the design and evaluation of a range of consumer products from mobile phones to video-recorders. They work with the software developers and industrial designers to explore the best designs for good interactivity.
- Usability consultants. Several organisations have recently emerged which offer a range of usability services to their clients.
- Software engineers with HCI training. Most computer science courses at universities throughout the world now include some instruction on HCI. The hope is that this education will bring awareness of the importance of good user-centred design and promote skills in people who are directly involved in building systems.

Activity 5 - Finding HCI research lab/ design specialists

Using the Web and a good search engine like Google find sites belong to the human factors/ HCI specialists in the following companies: IBM, Microsoft and Google. Spend sometime looking at each site.

- What types of work do they do?
- What kinds of people do they want to recruit?

Disciplines involved

HCI draws on a wide range of disciplines as we will see during this course. They include:

- Psychology
- Sociology
- Information systems
- Product design
- Computer science

Its natural ‘home’ though is computer science where it is a central concern.

Review Question 6

What contributions do you think that each of the following professional might bring to the field of HCI?

- Psychologists
- Sociologists
- Information systems experts
- Product designers
- Computer scientists

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

The Rest of this Course

This unit has been an introduction to the course. In the next 9 units you will learn and think about much more:

- Motivations for studying this course. We will look in more detail at why HCI is important and what might happen if you get it wrong.
- The capabilities and limitations of human information processing. You need to know about how humans process information (from their senses): how does visual processing work? What other senses are used? How does human memory operate? These and other questions will be addressed and you will explore how this knowledge can be used to build better interfaces.
- Interaction technologies. You will learn about the range of devices and methods available for humans and computers to communicate.
- Tools and techniques for user-centred design. This is an Important part of the course. You will learn about and practice using a range of tools that can be used to make sure that users' concerns are at the hearts of the design process.

Online Resources

There are a range of Web based resources that you should use during the course. There are many excellent Web sites with superb coverage of HCI issues. Specific reference to sites will be made in the units following this one but three good starting points are

- ACM Sigchi resources [http://www.acm.org/sigchi]

- Jacob Nielsen’s usability site [http://www.useit.com]
- Bruce "Tog" Tognazzini usability site [http://www.asktog.com]

You also have access to the ACM’s digital library. In this resource there are online versions of a number of key conference proceedings and research journals on HCI. You should visit each of those sites NOW to see what is available.

- ACM Interactions - A bi-monthly magazine on applied human-computer interaction.
- ACM Bulletin - Official publication of ACM SIGCHI, featuring columns, reports, articles and news on the subject of Computer-Human Interaction.
- ACM TOCHI - Transactions on CHI. Research journal on human-computer interaction.
- ACM CHI conference proceedings – premier annual conference on HCI.

Activity 6 - Using the ACM Digital Library

The ACM digital library [http://www.acm.org/dl] has a huge range of HCI related resources. You can browse specific publications or search across the range of journals, conference proceedings and other forms of literature.

- Go to the library now.
- Download the current version of the HCI magazine Interactions.
- Read some articles in this magazine.
- Return to the library’s homepage.
- Search for ‘speech recognition’.
- Select an article that interests you and read it.
- Experiment with the digital library – find out the range of search types and browsing you can do.

Summary

This unit has introduced some of the key concepts and issues you will be working with over the next weeks. You have seen what HCI is about and why it is important. You have also been given an overview of the rest of the course and seen all the coursework requirements for this course.

Answers and Discussions

Discussion of Activity 1

During the week you might have encountered a whole range of systems (mobile phones, cash machines, ticket machines, word-processors, the web etc) as you attempted to achieve a diverse set of goals (travelling from home to work, completing an assignment, finishing off your home accounts etc).

Some systems would have been usable and useful whilst others left you confused and frustrated. Some of the systems you might have used for the first time – what was your experience with these novel situations?

Discussion of Activity 2

What would a human travel agent do if you tried to select an outward flight from one airport and a return flight to another?

They would probably draw your attention to the possible problems/ errors. So, the online system could bring up a warning message to highlight the possible unintentional booking you were about to make.

Discussion of Activity 3

Sue’s characteristics might include:

Physical

- Frailty – inability to press down on ordinary ‘hard’ buttons.
- Wheelchair bound – ATM at wrong height.
- Poor eyesight – difficulty in reading screen and legends on keyboard.

Mental

- Memory problems – difficulties in remember PIN code and complicated sequences.

A redesigned ATM might include:

- Large screen with large fonts for readability.
- Large buttons with large legends. Easy to press.
- Use of eye-based/ fingerprint-based validation to avoid need for PIN code.
- One-to-two steps to get any transaction completed.
- Speech synthesis options?
- Speech recognition to avoid need for hand-based input? (what would the problems of using speech be?)

Answer to Review Question 4

Some favourites amongst previous students include:

- Trapped Between Doors [http://www.baddesigns.com/doors.html]
- Top-loading VCR [http://www.baddesigns.com/vcr.html]
- Where are the Stamps? [http://www.baddesigns.com/stamps.html]

Answer to Review Question 1

An interactive computer system is one which requires dynamic, frequent intervention by the user – online e-commerce sites, word-processors, ATM (cash machines) are all good examples.

Utility billing systems, bank cheque processing systems and direct mail letter printing systems are all examples of non-interactive system. These systems work repetitively through large amounts of data doing well defined and set tasks.

Answer to Review Question 2

'Ubiquitous computing' is a term that looks to the future when computing systems will be deployed extensively within the environments we live and work within. Many people believe that these systems will be all around us, sensing events in the locality and providing us with effective, focused information and services any time any where.

Answer to Review Question 3

A usable system is likely to be: learnable, flexible and robust. Users should be able to easily assess how they might use the system to achieve their goals and should be able to evaluate their progress as they use the system.

Answer to Review Question 4

Your aim is to convince the board that spending money on HCI makes good business sense. Ways of doing this would include showing how HCI might reduce costs or generate incomes.

Could argue that by using HCI methods that products would improve such that:

- Home users that bought the systems would be satisfied and feel good about their purchases. Their quality of life would be improved. This might lead to them buying further products in the future or recommending the company to their friends/ colleagues. Bad interfaces will lead to confusion which might mean that the company has to spend more on support (e.g. help lines or documentation).
- Business users might find that the company becomes more productive – look again at the notes on productivity and HCI. This will lead to potential further sales.
- If disasters/ fatalities occurred because of poor interfaces in the safety-critical systems, the company could be very seriously affected. It might be sued for high amounts of damage costs and its reputation would be greatly tarnished.

Answer to Review Question 5

Summative evaluations occur once a system has been completely developed. In contrast, formative evaluations occur during development. Summative evaluations are required in two cases:

- where a system has been developed in-house, a summative evaluation acts as a final check on usability and system acceptance. The system deployment should be delayed if problems are noted at this stage.
- an organisation may need to decide between several possible bought-in (off-the-shelf) solutions. Summative evaluations can produce comparative information from a user point of view (e.g. which one is easier to learn, fits best with employees working practice etc).

Formative evaluations should be used whenever a system is being developed – these evaluations should help produce improved designs as the system evolves.

Answer to Review Question 6

Psychologist – knowledge about the human capabilities and limitations in terms of information processing. How we perceive information (through our eyes, ears etc), focus our attention, use our memories, reason and problem solve.

Sociologist – knowledge about how people work and relate together. Particularly useful when designing the newer types of CSCW (computer support of collaborative work system).

Information system expert – various systems methodologies have been developed which attempt to involve users in the design process (e.g. Soft Systems Methodology).

Product designer – ergonomic aspects of the physical elements of the interactive system.

Computer scientist – knowledge about potential interaction technologies and methods; capabilities and limitations of the computer.

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.

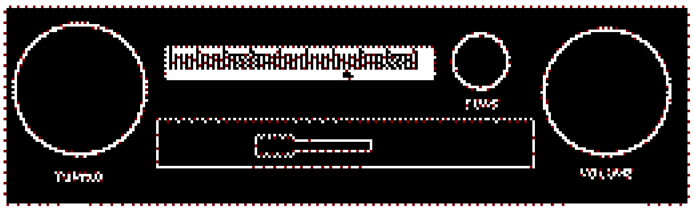
Efficiency: The resources expended in relation to the accuracy and completeness of goals achieved.

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

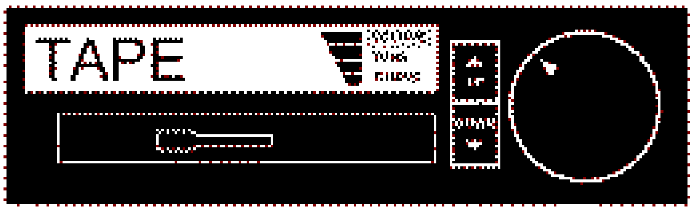
This is a rare example of a usability standard. It describes what ‘usability' is: a combination of factors, effectiveness, efficiency and satisfaction. Each of these are then sub-defined. This process of breaking a definition down into smaller more quantifiable units is popular, but contentious. This 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



An old fashioned in car



A modern in car stereo

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.

Chapter 3. User Centred Design

Table of Contents

Context

In the previous unit you explored various arguments for why user-centred design (or UCD) should be a central feature of interactive systems design. This unit will present an overview of the key UCD tools and practices and show how they relate to conventional systems development processes. UCD tools and techniques differ various ways including their viewpoint (e.g. cognitive psychology or computer science) , formality (e.g. compare evaluations by inspection with formal user modelling for instance), user involvement, when they are applied and deliverables.

Objectives

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

- Identify weaknesses in traditional software development methods in terms of user-centredness.
- Show how UCD methods can improve the design process for interactive computer systems.
- Describe what might make up a UCD toolkit – the methods and tools.
- Distinguish between the various types of UCD tool and technique.
- Know when to apply the tools/techniques

Introduction

In the previous unit we looked in a critical way at interactive systems and their development and did not like much of what we saw. In this unit we begin to look at more positive things: solutions.

There are many tools and techniques for designing systems for improved usability, and the question which should immediately come to mind is ‘which is the best?’ The answer is: ‘it depends’. There are many types of interactive systems and many types of users using them for many different tasks. The initial challenge for a developer wanting to make use of these techniques is to find the most appropriate one for the job at hand.

The first major distinction is between ‘design methods’ and ‘summative evaluation methods’. Essentially the distinction lies in when they applied to the interactive system:

- during the design process, i.e. before there is a tangible system that has been developed, or
- after the majority of the design process has taken place, i.e. when there is an implemented system that can be tested.

There is also a technique known as ‘rapid prototyping’ which aims to occupy the middle ground between design methods and summative evaluation.

We will later present arguments that design methods are preferable, as they should get the major problems out of the system before a costly implementation is produced. Summative methods may identify major problems, but it is likely to be extremely expensive to remedy big problems at this late stage in the design process. Summative methods are still necessary, but should ‘clean up’ small problems with usability such as screen design, while major issues such as fitness for task have been addressed earlier in the design process.

It should be a case of ‘either-or’ for a successful development scheme; to build really usable systems on time and to-budget many of the design and evaluation methods we will describe in this unit will need to be used, but crucially they need to be used well and intelligently by designers. There is not a panacea for usability; no one technique will guarantee a usable system. Many of the techniques that we describe are about identifying usability problems, and structuring the solutions to those problems. How the problems get solved is still left to a great extent to the skill and knowledge of the designers.

Review Question 1

There are two techniques for improving the usability of interactive systems: design techniques and summative evaluation techniques. What differentiates the two? Why, in general, is it impossible to evaluate a system for usability during the design process.

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

Activity 1

The task artefact cycle describes a common occurrence; namely that a developer produces a tool for a certain task, and then finds that the users of that artefact use it in a completely different way to that expected.

Think of two examples of where you have used a tool in a way that it really was not designed for, one example being a computer application and one from the wider world.

Now think of the tools/computer programs you use at work or home and see if you can think of ways in which you could use them in different ways to how they were intended.

A discussion on this activity can be found at the end of the chapter.

Traditional Software Development

In this section we look at some ideas and descriptions of how systems get designed. None of these processes of system development take much account of usability, but it is important to understand them so that you can understand the ways in which user centred design can be built into those processes.

Waterfall Development

The waterfall scheme for software development is the classic description of how software is (or should be) developed. There is debate as to how a good a model it actually is of the development process and plenty of variations on its theme have been suggested and described in many software engineering textbooks. Here we shall just concentrate on the classic waterfall model and its implications for user centred design.

Different schemes for development abound but the implications and challenges they set for user centred design are roughly the same.

The waterfall model of software development (see Figure in next section) describes the stages that a development project should pass through from initial contact with the customer and requirements capture, through design and programming to implementation and maintenance.

For the rest of this unit we make the following distinctions: the ‘customer’ is the party who determines what is wanted of the software, and usually pays for it, the ‘users’ are those that actually have to interact with the implemented software and the developers are those that design and implement the software. It is important to realise that the customer need not be the user.

Also note that software is rarely developed on its own, it is usually part of a wider system that is being implemented. Dix et al use the example of aircraft development. The overall lifespan of an aircraft project can be up to fifty five years, from initial conception to decommissioning. Designing and building software subsystems for an aeroplane is just one part of the many processes that need to carried out to develop and run an aircraft successfully. In this unit we concentrate on the development of software systems, not on overall system development, which is a much wider concern.

In the following description we use the design of a car as an example; it is not a software system, but we use it because it is something familiar.

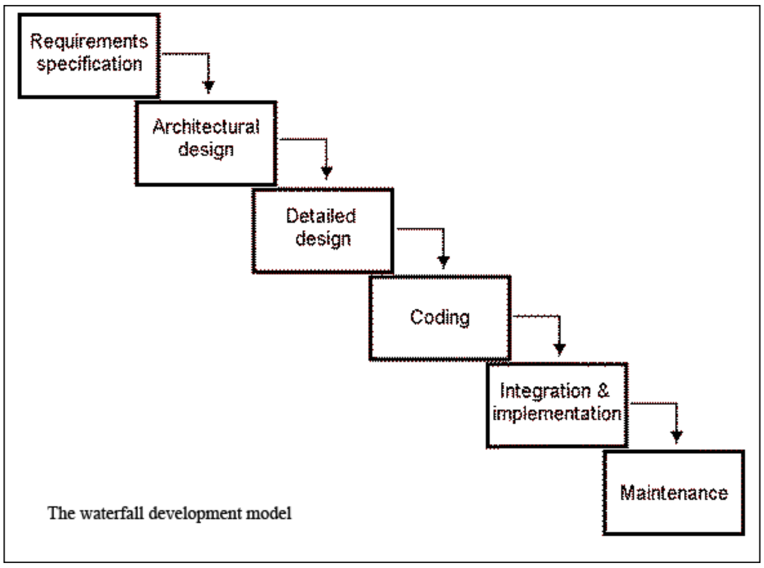
Now we step through each stage in the waterfall model, describing the activities that go on in each stage.

Requirements specification

Requirements describe what a system should do. It is important to distinguish the description of what a system does, from the description of how it does it, which comes later in the cycle.

The developers should liaise with the customers to determine what problem the software is intended to solve or what work it is intended to support. They should also collect information on the environment in which the system is intended to sit, how it relates to any other systems and its user population.

The requirements typically form a contract between the customer and developers. The developers will be obliged to produced a system which does everything that the requirements say it will.


Requirements specification
Architectural design
Detailed design
Coding
Integration & implementation
Maintenance
The waterfall development model

Requirements therefore form an interface between customers and developers. The requirements must be understood by the customers, so they realise what they are paying for, and requirements must also be specific and precise enough that developers know what they are supposed to be building. Effectively, to produce a good set of requirements the customer and developers need to be speaking the same language. This may sound a trivial stipulation, but it can be surprisingly difficult to achieve. there are many documented problems where developers produce exactly what is described in the requirements only to find that the requirements do not describe the system that the customer actually wants.

If we were designing a car at this stage in the process we would try to address the problem we are trying to design the car to solve: are we racing the car? or is it a commuter car? or an off-road vehicle? Each of these would give us different requirements for the car: fast, efficient, comfortable, robust, etc. Note that each of these requirements states what is wanted of the car, not how it will be built.

Architectural design

Given a set of requirements the developers can now set themselves to the task of actually producing software. It is a well known software engineering maxim that the effort required to decompose a problem into lots of smaller problems, solve those small problems and then reassemble those solutions so that they solve the big problem is much less than the effort of solving the big problem in one go.

Based on this principle the architectural design phase decides how to break the problem presented by the requirements into smaller, tractable sub-problems. There may very well be software components available to fulfil some of the sub-problems identified as part of the architectural design phase. Decisions will be made as to whether it is more cost effective to purchase this software or develop it from scratch.

The result of the architectural design phase is the specification of several subsystems that can be developed independently of one another. There will also be a description of how to reassemble these subsystems into the system that satisfies the requirements.

In the case of the car design the car will be split into subsystems: engine, transmission, bodywork, steering etc, with specifications set for each of those subsystems. Those specifications should reflect the overall requirements set in the requirements specification stage. The transmission for an off-road vehicle will have very different specifications to the transmission for a fuel economic car. Also at this stage decisions will be made about buying the components ‘ready made’; will the designers design and build a new engine, or will they buy in an existing make of engine from subcontractors?

Detailed design

If the architectural design phase was conducted well then designers will be able to develop the specified subsystems independently of one another. Indeed in large projects the responsibility for subsystems may be contracted out to other software houses to develop.

For the car, detailed drawings of mechanical components will be produced and detailed assembly guidelines will be put together. At the end of the detailed design phase there should exist an accurate engineering model of the car, from which the car can be built.

Coding

Once the design for a subsystem is completed then programmers can start to produce code. The design phase is now firmly dealing with how a system performs. Once coded the subsystem will be tested to ensure that it fulfils its specification.

There are several techniques for ensuring that correct code is produced. Given that the specification of a subsystem can be expressed mathematically and that program code can be given a mathematical semantics it is possible to mathematically prove that given code is correct to the specification. Doing so is rather arduous, highly skilled and therefore expensive. Only developers of safety critical systems, where correct functioning is absolutely crucial, usually go to these lengths.

Typically a subsystem will be coded and then tested. Generally though it is never possible to fully test a subsystem.

For example, assume that a developer is working on a subsystem that turns off a fuel valve in a boiler when certain conditions occur. The specification will describe the circumstances in which the valve should be shut off: when a certain temperature or pressure is exceeded, or when too rich a fuel mix passes into the boiler. The software subsystem takes input from three sensors which give measurements of temperature, pressure and fuel mix. Let us say that each sensor produces a thousand different possible outputs (a very conservative estimate). Then the subsystem will have a thousand to the power of three (one thousand million) different input configurations. It is simply not possible to test this many configurations, and this example is much simpler than most subsystems developed in the real world.

Mathematically proving that a subsystem fulfils its specification is very arduous, but for all but the simplest subsystems it is impossible to fully test it. Typically subsystems will be tested against crucial inputs (in the case of our boiler it would be tested for the extreme values; the values that move the system from safety into danger, where the valve must work.)

Obviously a car is not coded, but the analogous phase in the design of a car is the production and testing of the individual components that have been designed.

Integration and implementation

Once the subsystems have been coded and tested then they can be assembled into the final system. At this point testing of the overall system will take place and usually the customer is brought back into the process to verify that the emerging system actually matches the needs that they expressed in the requirements. Once the customer verifies this then the system is released to the customer.

Given the manufactured and tested components from the ‘coding’ stage these can now be assembled into a working car. The overall car now exists and can be tested. Of course what is interesting is the extent to which it matches the original requirements set in the first phase.

Maintenance

Once a product is released, work can still continue on it in situ. Indeed it is very unlikely that the developers will have got the system completely right first time and so maintenance needs to take place. It must be noted that the development of a substantial system may take a year or so from requirement capture to implementation, but maintenance may last the working life of the system; twenty years or so. Therefore the major cost of the development actually lies in maintenance. Developing the system carefully may be initially expensive, but should reduce the effort needed in maintenance and therefore pay for itself.

Cars still need servicing and looking after while they are running. Generally this is up to the owner to attend to, but the manufacturer can also be held responsible with warranties and such like.

Discussion of the waterfall model

The waterfall model was developed in the seventies in order to give a framework and identifiable process to software engineering. At this time most software systems were inventory type batch processing systems, with little or no interaction with users.

Now that most systems are interactive, the appropriateness of the waterfall model has been called into question. The waterfall model is intended to develop systems that are primarily functionally correct. Requirements and specifications typically describe what the system should or should not do, and the waterfall model is intended to give developer a process whereby they can produce software that satisfies those requirements.

‘Non-functional requirements’ are not so much about what and how a system operates, but the way in which it operates. Non-functional requirements tended to be treated as of secondary importance to functional requirements. A typically non-functional requirement would be the reliability of a system. If a system did what was required of it, excellent, if it did that without breaking down very often then, that was a bonus.

In the waterfall model usability is considered a non-functional requirement and is therefore given a secondary importance, but in an interactive system usability should be considered as important as function correctness. (Which is not to downplay the importance of functional correctness – the most unusable systems are ones that do not work.)

Later we will briefly look at attempts to tag usability onto the waterfall model, by adding an ‘interface design’ phase. We argue that this is insufficient for genuine usability to ensured and that a concern for usability needs to be addressed at all stages in the waterfall process.

Another interesting consequence of the waterfall model is the cost of making mistakes. If you make a mistake somewhere in the process you usually need to cycle back in the process by going back to the phase in which the mistake was made, correcting it and then continuing through the phases from there. As a rule of thumb it is usually cheap to rectify a mistake if it is spotted soon after it was made.

Hence if a mistake is made in the coding and it is spotted in the implementation phase then it is usually quite cheap to remedy. If, however a mistake is made in the requirements and it is not spotted until implementation then, in effect, you have to rebuild the whole system, which is very expensive. It is therefore absolutely crucial that the correct requirements are gathered, because the customer is only brought back into the process at implementation. If they identify a mistake then, or, more typically, realise that the system produced is not actually the system they really wanted, it is very expensive to remedy these problems.

The waterfall process is a structure to describe how developers should go about designing systems. The concept of design itself is interesting, and has its own particular terminology. In the next section we describe design and some of its terminology.

Designing

Design is about finding solutions to problems. Good design is about finding solutions that solve problems well. Given any problem there will be a myriad of different ways of solving it. These different ways are collectively called the ‘design space’. It is up to the designer to make a choice about which solution in the design space is best suited to the problem. A single decision is called a ‘design step’.

Also important in design is the notion of ‘discharging’ a design ‘obligation’. As a designer you are presented with a statement of a problem, which it is your job to solve. This statement of a problem is known as the obligation, (because you are obligated to solve it) and once you have solved a problem then you are said to have ‘discharged’ that obligation. So a design step is about devising a solution to a problem, and the step is held to be discharged successfully if the designer can demonstrate that their solution solves the problem.

Good design practice tends to put big decisions off as long as possible. If you think of the process of design as being the gradual narrowing of the design space until a single solution is arrived at, then a big decision at the beginning of the process will narrow the design space considerably and leave little scope for design decisions to be made later. ‘Never do today what you can put off until tomorrow’ is a popular and only half jokey maxim of good design.

Now if we think simplistically then for any problem there is a design space, and that design space can be partitioned into solutions which are usable, and solutions that are not. We obviously want to encourage designers to always choose a solution from the usable ones. The following approaches to user centred design are all about helping designers identify which solutions in the design space are usable and which are not.

Things are not that straight forward of course. Usability decisions are unlikely to be clear cut. So the following approaches sometimes also suggest what a designer should do in case of ambiguity and conflict.

The waterfall process is very rigorous and structured. In the next two sections we look at much less rigorous processes, in fact they are so unstructured it is difficult to describe them as processes at all. They are more rough approaches to development, or styles of development. While they are not structured or formal, they are quite common in the real world.

Review Question 2

Why is it cheaper to make mistakes and rectify them early in the waterfall design process?

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

Review Question 3

What distinguishes a design step that is good from one that is correct?

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

Evolutionary Development

Whereas the waterfall model has a beginning, middle and end, evolutionary development has no such concepts. A system exists and is added to, modified and tinkered with over time to improve it.

The motivation for the improvements may come from different places: new advanced technology may become available that the maintainers of the system may want to use, or new ideas about how the system can be, or is being used, come to light and are incorporated into the system.

Typically innovations are implemented in two ways: firstly small improvements will be added to the working system without taking the system off line. Secondly, radical overhauls to the system may be decided on. In this case the old system will be left working in situ, while the improved system is built and tested concurrently. When the maintainers are happy that the new system is working well then the old system is phased out and replaced with the new.

A good example of evolutionary development is that of digital library systems.

Traditional paper libraries have been around for a very long time. Gradually automation is taking them over. Catalogues have been automated for a long time, and users have been able to electronically search to see if a certain book or journal is in the library, see whether it is on loan and find out which shelf it is housed on. Internet technology means that such catalogues can now be viewed by users at distance. A user can sit at their terminal and see if a book is in the library without having to go there to find out. Then the ability to order books remotely was added to the catalogue systems; the user need never leave their terminals, they could order books from the library and have them delivered to their desk.

Recently publishers have begun to produce journals and books in electronic as well as paper format, and distribute them on CD-ROM and DVD-ROMs. Users can search those media in the library and obtain electronic copies of the books or journal articles they want. Electronic documents can be delivered over the Internet, so the user can simply sit at their terminals and read documents straight away.

The waterfall model is not really sufficient for describing this process. Requirements change with the possibilities offered by the system. A developer designing an electronic catalogue system ten years ago could not really have foreseen the possibilities for libraries that have opened up with the Internet. It is safe to assume that the developers of current digital library systems cannot predict the way that information delivery will change over the next ten years either.

Revolutionary Development

Lastly we look at revolutionary development. There is no set process to a revolution (by definition) and revolutionary software pops up now and again and changes the way we think about computerised systems.

The classic example are spreadsheet programs. The first was developed in the late seventies by a business student fed up of having to perform repetitious calculations. He developed an ‘electronic worksheet’ that did all the work for him. The concept seems obvious and simple now, but it revolutionised the way that many accountants work.

Revolutionary products come from a ‘eureka’ idea: a burst of creative thought that people are famously better than computers at producing. Eureka moments are not understood to any useful degree by psychologists and are therefore not really supportable or predictable.

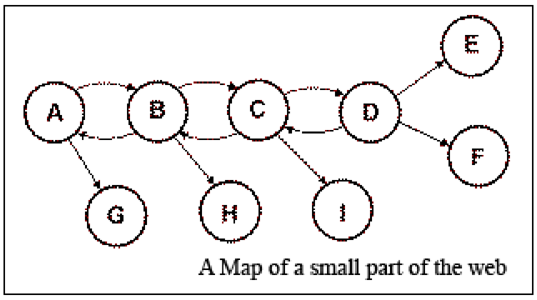
It is salutary to note that though a lot of developers spend their time trying to formulate the next ‘big thing’, the number of genuinely revolutionary computer products in the past twenty years number about five, whereas as the failures that tried to revolutionise are legion.

Review Question 4

Which of the cycles (applied science or task artefact) we described in the opening to this unit best captures evolutionary development?

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

Activity 2



Recall the analysis of web browsing you performed in unit 2. Now we shall proceed to use that analysis to redesign the web browser, hopefully in a more user centred way.

From unit 2 activity 6 should have resulted in a list of things that you actually did with your web browser, and activity 7 should have resulted in a list of things that your web browser lets you do. This activity compares the two.

Go through the list of browser functionality from unit 2 activity 7 marking off which of those functions you actually used in any of the tasks from activity 6. Apply the following scoring system: if you did not use a function score it 0, if you used it once or twice score it 1, if you used it several times (three to nine times) score it 2, if you used it a lot (over ten times) score it 3.

Now step back. How much functionality scored 0? Most of it? How many functions scored 3? I would guess that the Back' button scored a 3, and maybe the bookmark menu, if you use bookmarks. (Some users do not use bookmarks at all and so they would score a 0).

Now conduct a similar analysis, but this time score each function just with a single task from unit 2 activity 4. So score each function for searching, and then for browsing, and for any other task that you set your self. Now compare the scores for each different task. Do you use different functions for different tasks? I use my bookmarks a lot when searching and the Back button a lot when browsing, but not vice versa.

If you studied other users compile scores for each of them and see how they differ from your scores.

You may have completely different results to me, but one thing should stand out most of the functionality offered by the browser is not used. In other words web browsers are not very well suited to the tasks to which users put them.

In particular lets look at the Back button: most users make great use of it, but can you explain what it actually does? Consider the map of a small corner of the web shown in figure 4. Each circle represents a web page and each arrow a link from one page to another. Imagine you start at page A and jump to B. On B you notice that there is a link to page H that looks interesting, but you first want to look at page C, so you jump there and then jump to page D. Now you decide that you want look at page H, so you need to backtrack to page B, this you do and then jump to page H. Now you press the Back button twice, where do you finish up? Page A or page C? Try and justify your answer.

So we have an even more damning argument for the usability of the browser; not only is most of its functionality unused, what the most used part of the functionality: the Back button actually does, is not very clear.

In summary we have shown that there is considerable scope for redesigning a web browser. This is what we shall look at in activity 4.

A discussion on this activity can be found at the end of the chapter.

User Centred Design

So far we have said little or nothing about usability. The previous section described how software systems get built, but not how usable software systems get built. What would be appealing to software developers would be a new box in the waterfall model called ‘design for usability’ in which the software or designs they have so far developed are passed along to a set of usability experts, who perform their own special rituals to the designs and pass them back into the process, guaranteed to result in improved usability.

The idea of user interface management systems (UIMS for short) is one way of appealing to this idea. UIMS assume that functionality can be separated away from the interface, so software developers can get on with developing their functionality while not worrying about how users make use of that functionality, while interface experts beaver away at designing interfaces that make that functionality easy to use.

Its a beguiling idea, but misguided. Consider the following example:

The Mini is a classic small car, revolutionary in its day and affectionately remembered by most people who owned one. Early models were very spartan in design and manufacture. The speedometer, fuel gauge and other indicators were housed together in a single circular unit which was placed centrally on the dashboard. (Whether this was to make production costs cheaper because the unit stayed in the same place no matter if the car was left or right hand drive is not clear, but it seems likely.) Unfortunately if a moderately tall driver were to use the car they would find that the hand gripping the steering wheel when the car was moving straight forward almost entirely blocked the view of the speedometer and other gauges. To see the gauges the driver would have to take his hand off the steering wheel; not a recommended activity.

Now given that design, the car’s manufacturers could have brought in the most skilled designers to improve the readability of the gauges, but it would not have made the slightest difference. The most well designed gauges in the world are useless if they cannot be easily seen.

The parallel for usability design is clear. If the functionality is inadequate or inappropriate for a given task, the most well designed interface is not going to mitigate that fact at all. Good user interfaces can make good functionality usable, but the functionality needs to be designed for the user too. Therefore user centred design needs to permeate through all the steps of the waterfall model.

The following is very important and should be taken to heart by all usability experts:

Good usability takes more than a good user interface.

Note however that the equating of usability with interface design is deeply ingrained. Even one of the recommended texts for this course seems to make this fundamental error from the outset: Shneiderman’s ‘Designing the user interface’. (This not to say that there is not a wealth of useful information in Shneiderman’s book, but it has an awful title.)

Guidelines

A guideline is a rule about designing interactive systems. They range from very general (‘Keep the user informed about what the user is doing’ from Nielsen 1993) to the very specific (‘Use white space between long groups of controls on menus or in short groups when screen real estate is not an issue’ from the Open Look style guide).

Shneiderman’s ‘Designing the user interface’ is to a large extent a collection of design guidelines, but explanations are included to show why those guidelines are the way they are. (For example look at box 7.2 on page 264 which gives a list of form filling guidelines and then the text below which explains the reason for those guidelines.)

There are two components to a guideline: the guideline itself and the rational behind it. The ultimate idea being that if the guidelines are sound and unambiguous enough then they could be presented to designers without the rationale behind them. The designers could then apply the guidelines, almost automatically, and be sure of coming up with an usable system.

Such an approach has precedence in all sort of situations; we use sets of rules and guidelines without needing to know the theory that underlies them frequently. A cookbook describes how to make an omelette and if its rules are followed then a perfectly good omelette will be arrived at. The cook does not need to know the science of what happens when the proteins in the egg white are heated in order to successfully cook an omelette.

Rationales for guidelines come from different places and can therefore give the guidelines different levels of authority. Much academic study has attempted to link guidelines to sound scientific theory and if such a link can be established then the guideline will have a considerable level of authority. Other sets of guidelines are based around common sense thinking or possibility previous experience; a practitioner will write down what worked for his design, possibly with some explanation of why.

It is important to distinguish between ‘standards’ which we discussed in the previous unit and the guidelines we are discussing here. They are both rules about what designers should or should not do, but standards carry much more authority; they have to be obeyed. Guidelines are much looser and are often transgressed. Because of the higher authority of standards then they must have a well defined and inspectable rationale behind them.

Problems with guidelines

Unfortunately the science behind usability is not sound enough that this sort of reliance on guidelines can be usefully achieved. The designer will in many cases need to understand, to some extent, the science or rationale underlying the guidelines so that they can be intelligently applied. In many cases usability guidelines contradict one another, so it is important to understand their rationale in order to decide which is the most appropriate.

Usability Engineering

Usability engineers explicitly set down criteria for their designs and then describe measurement schemes by which those criteria can be judged.

An example: Advanced mobile phones pose interesting HCI challenges. Mobile phone manufacturers want to cram as much functionality into their products as possible, but the physical size of the phones and their displays means that normal features for accessing lots of functionality: menus, dialogue boxes, etc., tend to be inappropriate. Assume you have to design a way of accessing phone functionality and you have decided on a restricted menu system. Menus must not be long, perhaps a maximum of six items per menu. To compensate for this you may decide to nest menus, but if you do this then it is easy for the user to forget whereabouts in the menu hierarchy they are, so feedback must given.

As a usability engineer you set yourself the explicit goal of avoiding ‘menu lostness’; the extent to which the user gets irredeemably lost in a menu hierarchy. On each menu there is the ability to ‘bail out’ and return to the highest level menu. If a user bails out like this without invoking some functionality then we assume that they have failed to find what they are looking for and become lost.

The usability specification for this aspect of the menu system may look like the following:

Attribute	Menu lostness
Measuring concept	success in finding and invoking the desired functionality
Measuring method	The ratio of successfully found functionality to bail outs
Now level	This is a new product: there is no recorded now level
Worst case	50% of menu use results in bail outs

Planned level	10% of menu use results in bail outs
Best case	No bail outs

The attribute ‘menu lostness’ describes what is wanted of the system in usability terms. (Note though that ‘menu lostness’ is a bit of an opaque term. There should be accompanying documentation described exactly what it means.) The measuring concept describes what we are looking for in a system in order to judge whether the attribute is fulfilled or not, and the measuring method describes how we should go about looking for it. The now level describes the current state of the system; how the current system fares against the measurements. The worst case, planned level and best case describe what the designer should aim for in their design.

Note that like good specifications this describes what is wanted from the system, not how it is done. It is up to the designer to build in features that prevent the user from getting lost. Whether they do this by adding explicit feedback on the display which describes exactly whereabouts in the menu hierarchy the user is, or by flattening the menu hierarchy somehow to make it harder to get lost, is not an issue for the specification. The specification describes what is wanted in usability terms and how to tell whether or not it is achieved.

Because this scheme is based on normal engineering practice it should fit well into standard engineering practice like the waterfall model. In such an augmented practice as well having to discharge design obligations about the functionality by showing that a solution is correct for its functional specification, the designer also has to discharge the usability specification by showing that it has been solved.

Problems with usability engineering

The problem with this sort of usability specification is that it may miss the point. What really is at issue with the mobile phone example may be not that the user cannot get at all the functionality easily, but whether the functionality that has been crammed it is really useful in the first place. Usability engineering must start from the very beginning of the design process to ensure that account is taken of these issues.

Also, it is not clear that what is specified in the usability specification actually relates to genuine usability. In the example given we decided bailing out was a measure of failure. What if the user is not aware of the possibility that they can bail out? Novice users may struggle on, getting more and more lost and frustrated, unaware that they can just bail out. By the measurement scheme suggested this behaviour would not count as a failure. Furthermore, and rather flippantly, the designer could ensure that no user ever bails out by removing the bail out function altogether. This may sound silly, but designers have been known to do this sort of thing.

Activity 3

Consider the usability specification we have given for the mobile phone menu system. It describes what the problem is, and how we will know that we have solved it, but not the solution. Have a think about the problem and see if you can think up some solutions. We will return to this in a later review question, so make notes and keep hold of them.

A discussion on this activity can be found at the end of the chapter.

Rapid Prototyping

There is a mismatch between designing systems and testing them with users. In the waterfall model a tangible system only appears towards the very end of the process. For most of the process the system exists only as requirements documents, design specifications and ideas in the designers’ heads. The absolute test of a system for usability is to give it to collections of users and see what they do with it. Unfortunately an actual system only exists at the very end of the design process, therefore user testing can only take place at the end of the process and, as discussed earlier, if a big mistake is identified at the end of the process then it is extremely expensive to fix.

Rapid prototyping is a process whereby mock-ups or prototypes of the system are produced all the way through the design process. These prototypes can be then given to users in order to judge their usability. User responses to the prototypes can be judged and feedback passed into the design process to guide the design of the actual system.

Different forms of prototype

Prototypes can be constructed in several ways ranging through levels of interactivity that they offer to the user.

Storyboards are the simplest form of prototype. They are simply rough sketches of the system and its user interface. User are guided through the system by analysts who show the users how they are expected to use the system, and show them what the system is expected to do. The analysts record user responses to the system and feed these back to the designers.

Storyboards may be very simple drawings of the system done on paper, or there are more sophisticated tools available which allow the analysts to create storyboards on a computer using graphic packages and apply some limited animation to the graphics. These allows a little more reality in the storyboards, but in effect the analyst is still in control and steps the users through the system.

In order to give the users more the idea of interaction then limited functionality simulations may be created. These are effectively mock ups of the system, made to look like possible designs of the finished system but with a much restricted set of system functionality behind them.. There are several packages which allow the rapid development of prototypes. They allow the positioning of various interaction objects (buttons, menus, slider bars, etc) on the screen and the attaching of simple behaviours to those objects. Such prototypes give the user the impression of interacting with the full system. The simplicity of the prototypes mean that they can be rapidly reassembled in differing layouts according to user feedback.

Simulations also allow for hardware designs to be cheaply evaluated. Something like a video recorder has built in hardware buttons and controls. Mocking up a physical version may be quite expensive, but a ‘soft’ mock up may be created on a computer, where a picture of the proposed control panel is generated and the user can interact with it using the mouse to ‘push’ the buttons.

Another technique for prototyping is the ‘Wizard of Oz’ technique where the user is given a mock up of the system, which actually contains little or no functionality, but is remotely linked to an analyst who pretends to be the system and makes it respond to the user input appropriately. Such a technique is cheap to set up and very cheap to change, because the analyst just pretends that the system behaves in a different manner. There are ethical issues with misleading subjects in experiments in this way though.

There are also a collection of ‘executable specification’ languages which allow designers to describe software specifications formally in a specification language. These specifications can then be rapidly and automatically converted into working programs, to which interface elements can be attached and a working prototype can be presented to the user. The automatic translation from specification to working program means that the working program will be very rough and inefficient; it requires human software developers to produce really good code, but computers can generate code from a specification that does what it required of it, but in a very inefficient way. The problem with executable specification languages is that they tend to require that the specifications are written in a certain way so that they can be made executable. Non-executable specification languages do not impose such restrictions on developers and are therefore more flexible and powerful tools.

Iterative design

The idea of rapid prototyping is that a mock up of the system is produced, tested, the results of the test are fed back to the developers and the mock up is thrown away. Iterative design pushes the philosophy a little further by relying on mock ups and prototypes, but using those as actual artefacts in the design process instead of throwing them away. An iterative design process collects requirements and then endeavours to produce a tangible, but very crude approximation of the system as quickly as possible. This approximation can then be user tested and then redesigned accordingly. More detail and refinements can be added to this system and tested as they are added.

Gradually a fully featured, working, and hopefully usable system emerges.

Problems with prototyping

The main problem with prototyping is that no matter how cheaply designers try to make their prototypes and how committed they are to throwing them away, they still have to make design decisions about how to present their prototypes. These decisions can then become ingrained into the final product. There is a factor to consider called ‘design inertia’ which describes how designers, once having made a design decision, are rather reluctant to relinquish that decision, admit it is wrong and change it. This is another reason why good designers put off big decisions as much as possible, knowing that if they make a small mistake early on it is easier and cheaper to rectify than a big early mistake.

Rapid prototyping is about exposing bad design decisions as soon as possible after they have been made. Guidelines and usability engineering are more about trying to get the designer not to make mistakes in the first place. Design inertia is therefore much more prevalent in rapid prototyping development than it is in other user centred design processes. Because prototypes do not get thrown away in iterative design processes then design inertia is even more prevalent.

Furthermore in spotting usability problems by user testing, the designer knows that there is a problem, but not necessarily what causes that problem, or how to fix it. Rapid prototyping identifies symptoms, not illnesses and not cures.

Review Question 5

Rapid prototyping differs in one crucial respect to design by guidelines and usability engineering. What is it? (Hint: refer back to your answer to review question 1.)

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

Activity 4

Now lets consider the way that the functionality of the web browser is put on the screen, particularly the buttons along the top of the browser. Look at the space they take up and compare that to how much they actually get used. then think about what is really important about the web. Go back to step one and look at what you wrote about the web: what is important about the web? I wrote that what was really important was the information in the web: it is the web pages themselves that are important not the navigation tools offered by the browser.

So now look again at the space used up by the buttons and how much space that leaves for showing the web pages themselves. Now think about the television analogy: how much space is taken up by the televisions navigation tools (i.e. the channel changing buttons) and how much space is taken up by the screen itself. Look at a typical web browser display. Typically, the navigation buttons occupy about a fifth of the screen, and our analysis shows that only one of those buttons gets any significant use!

Now lets stop being destructive of the web browsers design and start thinking of constructive design ideas. We should have a picture of what functionality the user actually uses and in what situations from activity 3. So using a pen and paper sketch out an interface design which gives prominence and ease of access to the commonly used functions, makes the viewable window with the current web page in it as large as possible and buries unused functions away in menus.

A discussion on this activity can be found at the end of the chapter.

Design Rationale

Design rationale is about explaining why a product has been designed the way it has. Throughout this section we have been describing ways of supporting a designer in making design decisions, i.e.

selecting one design out of the design space. For each decision made there must a set of reasons why that particular decision was made. Design rationale is about recording those decisions and the reasons why they were made.

A design rationale is a useful design tool because it explicitly lays out the reasoning behind a design process and it forces designers to be explicit about what they are doing and why they are doing it. In particular a design rationale can be used after a product has been completed in order to analyse why a product was a success or failure. If a similar product is being designed subsequently then its designers can refer to a design rationale to discover why earlier products were designed the way they were, and with the benefit of hindsight judge whether the earlier design decisions were successful and warrant repeating.

Design rationales are particularly helpful in interactive system design because, as we have been discussing, there is rarely one objectively correct solution to any problem, and some solutions may contradict one another, or require trade-offs. Design rationales require the designer to be explicit about how contradictions were resolved and trade-offs were made.

Furthermore the design space may be very large and therefore it is not obvious that a designer will even consider the best solution, never mind choose it. A design rationale makes it clear which options from the design space were considered and why. If an apparently better solution were later discovered then it is obvious whether that solution had been considered and discarded for some reason, or not considered at all.

Usability is very context dependent; what is good for one user may be dreadful for another. If subsequent designs are made where the context of use does not change then a design rationale can be reused without modification. If however the context does change then new design decisions can be made for this new context, but in the light of the decisions made for the older context.

The QOC (Questions, Options, Criteria) analysis technique is a design rationale. It is a graphical notation that allows designers to visibly lay out the reasoning behind design decisions.

Questions are effectively problems in the terminology we have been using. Options are possible solutions to those problems, and criteria are arguments as to why or why not a given option is appropriate.

Note

Returning to the mobile phone menu problem we discussed in the usability engineering section, a question may be ‘How to prevent users becoming lost in menus?’

Options for this question could be:

1. to flatten menu hierarchies to a maximum depth of three,
2. to give textual feedback as to the current menu display (e.g. display the text ‘Address book : Edit : Add number’ to show that the user has selected the ‘Address book’ option from the main menu, the ‘Edit’ option from the Address book menu, and ‘Add number’ option from the edit menu.
3. to give graphical feedback as to how deep the user is in the menu hierarchy by shifting each new menu selected a little down and to the left, giving the impression of stacking up menus.

For each of these options there are several criteria effecting whether or not it is a good design decision:

1. Screen real estate
2. Limited functionality
3. Accurate user knowledge as to where they are in the hierarchy

Now we can go through discussing each of these criteria for each of the options.

1. If menu depth is kept to a maximum of three then:
   - this has no real effect on screen real estate,
   - it limits the functions that can be placed in the phone. Say a maximum of six items per menu can be displayed, then there is a maximum of 258 functions that the phone can perform,
   - keeping the menu hierarchy flat, improves the chances of the user remembering where they are.
2. If textual feedback is given then:
   - If textual feedback is given then:
   - there is no limit to the number of functions,
   - user knowledge of where they are will be very accurate.
3. If graphical feed back is given then:
   - screen real estate can be efficiently used,
   - there is no limit to the number of functions,
   - the user will have a prompt as to where they are in the hierarchy, but not a very accurate one.

So for the question we have identified three options and three criteria which effect those options.

Now it is up the designers to argue which is the best design option.

Review Question 6

Imagine you are the designer faced with implementing one of the options. Which would you decide to implement and why? Is this a good analysis of the menu design space? Explain your answer.

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

Modelling Techniques

In later units we will deal in considerable detail with modeling techniques, but we shall give an overview and introduction here. Generally models are approximations of real world artefacts which can be analysed. A good model is one which fairly accurately predicts the behaviour of the real world artefact it represents while doing away with a lot of the confusing complexity. Models are therefore tools for ‘abstraction’. What is important for an analyst using a modeling technique is that it abstracts away the irrelevancies and keeps the important facts. A model that does so it called ‘well scoped’, it is up to the analyst to chose a modeling techniques that is well scoped for the questions they want to ask of it.

Task analysis

A task analysis technique makes a model of the job that a user is expected to perform. Analysis techniques can be applied to those models in order to determine such facts as how long it may take users to perform given tasks, or how much ‘cognitive load’ is placed on the user (where ‘cognitive load’ is broadly a measure of much information the user needs to remember).

The most researched task analysis technique is GOMS (Goals, Operations, Methods and Selection) developed by Card et al (1983). The analysts describes:

- the user’s goals, or the things that user wants to achieve,
- the operations, or the things the user can do, perhaps such things as thinking or looking, or maybe selecting items from computer menus, typing or pointing with the mouse,
- the methods, which are sequences of operations that achieve a goal, and
- selections, which describe how the user may choose between different methods for achieving the same goal.

Once this model of the user’s task has been compiled then measurements of the time it takes to perform the operations can be added (these measurements are taken from empirical observations of users) and a prediction of the time it will take the user to perform a task can be calculated.

GOMS only really deals with expert behaviour though and takes no account of the user making errors. It has been argued that a surprisingly large amount of user time is spent making, or recovering from, errors, even for expert users. Therefore the accuracy of GOMS models have been questioned.

GOMS analysts scored a notable victory however when they accurately predicted that users of a new computerised telephone system would take longer to perform their tasks than users using an older, apparently slower system (Gray, John and Atwood. 1993).

User modeling

Whereas task analysis aims to model the jobs that users do, user modelling aims to capture the properties of users. User models can capture and predict properties such as the way the user constructs goals, how users make plans to carry out those goals, the users’ ability to do several tasks at once, how the user manages perception, etc. Such models are based to a large extent on psychological theory, which in turn is based on empirical evidence.

Typically the analyst builds a model of the interactive device that is intended to be built and then integrates this device model with an existing user model. This integrated model will be able to predict certain behaviours, and the analyst can therefore gain an idea as to whether the user will be able to reasonable perform the tasks that the analyst wants them to.

Interactive device modeling

Several techniques have taken existing software specifications and analysed them ‘from the users’ point of view’. In other words the analyst takes a model of the interactive device, which is typically produced as part of the design process, and analyses it for ‘usability properties’. A lot of work went into proposing usability properties and then formalising mathematical equivalents of them. In this way software specifications could be mathematically analysed for usability in much the same way as they can be analysed for functional correctness.

Dix’s PIE model (Dix 1991) is a classic example of interactive device modelling. The device is modelled as a collection of states with allowable transitions between them. This model can then analysed mathematically for such properties as ‘reachability’; the ability of the user to get from such state to any other allowable device state. Dix also formalised what it means for a system to be so called ‘WYSIWYG’ (what you see is what you get) by separating the displayed output from the printed output in his model and mathematically showing correspondences between the two.

Problems with modeling

As we explained above, models must be used with care. Because they abstract details away the analyst must understand what is being abstracted away and be able to argue why those details are not important. Furthermore modeling is seen as a highly skilled and time consuming activity. The idea of modeling is to be able to predict usability issues before a system is built, but many developers have the impression that modeling can be so complicated and highly skilled that it is cheaper to just build the system and then test it on users. Refutations to this are rare, but compelling (e.g. the GOMS analysis of the phone system we alluded to above). We will return to these issues in much more depth in subsequent units.

Activity 5

Activity 5 winds up the web browser analysis and redesign process by discussing what we have done. Consider my redesign for the web browser, along with your redesign, and the original design of the browser and compare all three. When there are differences between the three try to argue which is the best' and, most importantly, why it is the best. Use the evidence you gathered in unit 2.

If the original design of the web browser is not good, try and think of why this may be. The designers working at Netscape and Microsoft are not stupid and do not inflict bad designs on their users wilfully, but if something has gone wrong with their design then there must be a reason for this. Hopefully we have developed designs for web browsers that are user centred, if Netscape and Microsoft come up with different designs then this means that they may be working with different priorities we are in this tutorial. What are those priorities, and why do they take precedence over usability. Consider the arguments about feature accretion we discussed in the Contents notes. Do Netscape's and Microsoft's products show signs of feature accretion? If so, why? And do you consider this to be a bad thing? Why?

You will be invited to discuss your analysis in the discussion section.

A discussion on this activity can be found at the end of the chapter.

Evaluation and User Feedback

Landauer states that user evaluation is the ‘gold standard’ for usability. In the previous section we mostly discussed ways that designs can be analysed for usability, largely in the absence of users themselves. Landauer argues that methods for predicting usability are all well and good, but the only real way for evaluating usability is by giving users finished products and analysing how they interact with them.

Evaluating user behaviour is one of the most advanced and well researched fields in HCI.

The classic example of designing with user feedback is the IBM 1984 Olympic Messaging System (See Landauer for a detailed description).

IBM had to quickly develop a messaging system for use by the competitors in the 1984 Olympic games. Because of the huge diversity in users (different languages spoken, different levels of IT competence, different expectations of the system, different cultural backgrounds, etc) there was no way that the designers could accurately predict the usability of the system before the athletes arrived at the Olympic village and started using the system. Furthermore the games only lasted for a few weeks, so there would be no time to correct the system during the games; it had to be right first time. The designers therefore conducted initial user studies with passers-by at the IBM research centre, followed by more extensive trials at a pre-Olympic event with competitors from sixty five countries. The system was then run on a large scale with American users before the opening of the games. Each of these tests identified errors with the system and designers did their best to fix them. The final system that was used at the games was robust and was used extensively without major problems.

Ways of Evaluating User Behaviour

There are many ways of evaluating user behaviour and we shall discuss in more depth in later units. Which techniques should be used depends on what information you need to get from the users. The sort of information you can gather ranges from quantifiable measures of performance time to much more qualitative aspects such as levels of user satisfaction. Which measures are taken and acted upon is dictated largely by the purpose to which the system is to be put. Designers of consumer leisure products are going to be much more interested in levels of user satisfaction than performance measures. However designers of systems where profits rely on performance and whose users are paid to use the systems are going to be much more interested in performance measures. Gray et al (1993) give an interesting economic figure: they were involved in the redesigning of the workstations used by New England telephone operators. Given the number of operators and the number of calls they took, Gray et al estimate that a one second reduction in work time per call would result in a saving of $3 million per year.

The cost of performing an evaluation can also be a very important factor; performing an evaluation and analysing the results can be very time consuming and costly. Some techniques for evaluation (e.g. questionnaires) are much cheaper than others.

Laboratory set evaluation

This method of evaluation derives from scientific psychological studies. Psychologists attempt to study behaviour experimentally by giving subjects tasks to do in a controlled environment. By controlling the environment psychologists attempt to control variables that may effect the subjects' behaviour. If the experimenter can argue that they held all variables steady except one, and that changing that one variable changes the subjects' behaviour, then the experimenter is in a good position to argue that the one variable has a causal effect on the subject's behaviour.

Laboratory set evaluations are intended to produce results that have a high level of scientific rigour to them. Typically users will be invited to perform certain tasks using a system under laboratory conditions, then other users may be invited to perform the same tasks using a variation on the system. If the two sets of users behave in different ways then the experimenter can claim that the difference in behaviour are caused by the differences in the system.

Users' behaviour can be recorded by videotaping or by programming the system to automatically record what the users do in log files. After the experiment the analysts will usually try to get more qualitative responses to the system from users by asking them to fill in questionnaires or by interviewing them.

The main criticism levelled at laboratory set evaluations is that they lack 'real world validity'. Users may behave in a certain way in the rather unusual setting of a laboratory, but there is no guarantee that they will not behave in a completely different way in the 'real world'. Furthermore care needs to be taken with the users that are asked to perform experiments. Many psychological experiments that are reported widely are undertaken in universities by researchers who use the undergraduate population as subjects. On deeper investigation you will find that psychological statements about behaviour do not really apply to the population as a whole, only to a rather small and demographically strange set of university students. Evaluators of interactive systems should take similar care to evaluate their systems with the sort of people who are actually going to be using it.

Laboratory set evaluation is a way achieving scientific rigour in an evaluation, which is a very strong requirement for an evaluator to set themselves. Unless the evaluator wants to publish their results in learned journals there are easier and cheaper ways of getting data about user behaviour. Furthermore laboratory investigations require skilled evaluators and specialised equipment in the laboratory; they can be very expensive.

Ethnographic studies

An ethnographic study is a way of getting round the problems of real world validity presented by laboratory set evaluations. Ethnographic studies realise that valid user behaviour is not to be found in laboratories but in the users' homes and workplaces. Ethnographers also realise that users behaviour is also influenced by the presence of the experimenter. Hence the experimenter will try and become part of the users' environment. If a study is being made of a system in a workplace then the experimenter will join the workforce and perform the tasks that the users do.

Because the experimenter cannot control the environment to anywhere near the same extent as can be done in laboratories then it is much more difficult to make claims of cause and effect. Ethnographic studies are a newly emerging way of studying behaviour and are beginning to gain respect for the real world and valuable insights they give into behaviour. New sets of procedures for performing ethnographic studies are emerging and as a field ethnography is rapidly moving towards scientific respectability.

Much care must be taken with ethnographic studies though; they must be studies of the users' behaviour and not of the experimenter's. Because the barriers between users and experimenters are deliberately broken down there must be good evidence in the evaluation that the experimenter is reporting the users' behaviour and not their own. Although an explicit laboratory is not required, ethnography is still expensive in terms of experimenter skill and time.

Questionnaires

Questionnaires are a cheap way of gathering opinions from a large number of users. They range in how prescriptive they are in what sort of answers the user can give. They can ask open questions and leave the user space to respond by writing free text, or they can give very specific questions with a set range of answers that can be ticked. The answers on questionnaires can be read automatically if they are 'tick box' answers as opposed to free text. Being able to read answers automatically can also dramatically decrease the costs involved in the evaluation. Web pages can also be written in the form of questionnaires so that users can automatically send information to the developers.

There is a trade off between the amount of freedom given to the user in how they fill questionnaires and the time and effort required to analyse the questionnaires. A questionnaire made up entirely of questions with a set of answers that the user must tick allow very little freedom, but are very quick and easy to analyse. The more you allow users to fill in free text, the more effort is involved in analysing them.

The questions need to be carefully written so that users can understand them and give useful answers. Users will soon get bored and fill in fallacious answers if they do not understand the questions or find them irrelevant. Furthermore questionnaires should be fairly short and to the point to prevent users getting bored. If a lot of questions are to be included then it is best to put the important ones first so that users answer them before giving up. Because filling in questionnaires is such a boring task for most users evaluators will often offer incentives to users to complete the questionnaires.

User interviews

Interviewing users is quite skilled; the interviewer needs to be able to get appropriate information out of the users, while leaving the interview open enough to let users add their own opinions, while not letting the users 'run away' with the interview and discuss things of interest to them, but not much to do with the system being evaluated.

Summary

Each of these approaches to evaluation have their own weaknesses and strengths. A really extensive evaluation will make use of many, if not all of these techniques in order to try and maximise the benefits of each. A small evaluation will probably get best results by conducting several user interviews; they tend to generate the best quality information. Although evaluating by questionnaire is appealing because of its cheapness, it should be applied with care; a badly designed questionnaire can generate misleading results.

Review Question 7

Each of these approaches to evaluation have their own weaknesses and strengths. A really extensive evaluation will make use of many, if not all of these techniques in order to try and maximise the benefits of each. A small evaluation will probably get best results by conducting several user interviews; they tend to generate the best quality information. Although evaluating by questionnaire is appealing because of its cheapness, it should be applied with care; a badly designed questionnaire can generate misleading results.

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

Problems with Evaluation and User Feedback

The main problem with user evaluation in designing interactive systems is that information about what, in usability terms, is wrong with a system comes very late in the design process. Recall that we argued that rectifying mistakes becomes more and more expensive the later in the design process they are identified. This means that usability issues identified by evaluation can be very expensive to remedy. Indeed in a lot of cases, too expensive to remedy and the product will get shipped full of usability bugs.

There is plenty of evidence of developers identifying problems by user testing, realising that it is now too expensive to fix them, papering over the cracks and shipping the product. A very popular way of papering over cracks is by creatively writing the user manual for the product. If a usability problem is identified then the developers can usually think of a way of getting around the problem, though this get around is usually complicated and difficult. The problem and the get around can be described in the manual, and to a rather trivial extent, this ‘solves’ the problem. Usability expert Harold Thimbleby claims that the usability of a product is inversely proportional to the size of its user manual; the bigger the manual, the less usable the product.

Many of the criticisms that can be aimed at rapid prototyping can also be aimed at evaluation techniques. In particular evaluation techniques identify where a system is unusable, but not why.

Review Question 8

Why is identifying a problem and a get around' solution and then describing that in the user manual not a sufficient way of solving' usability problems.

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

Guidelines for User Centred Designing

In this unit we have given a broad outline of the techniques available that can be used to improve the usability of an interactive system. The question is: which technique is best? The answer is: it depends. The most important thing to realise is that there is no usability panacea. To be successful a developer must understand what techniques are good at and what they are not good at.

For example task analyses are good an analysing systems where the user has a set, explicit goal that they will try to achieve. They are not so good at analysing systems where the user acts in a more exploratory way. Task analysis would therefore not be very good at analysing web surfing systems, but would be much better at analysing an accountancy package.

The ultimate test of usability for a system is to analyse how users behave with it. What we will promote in this course is the idea that developers can design with users in mind, and therefore improve usability before getting to the evaluation stage. We are, however, not advocating designing interactive systems in isolation from users.

One of the main themes of this unit has been that making changes to system designs is fairly cheap and painless, whereas making changes to finished products is very expensive. Ideally we would recommend a design process where usability and usefulness are considered throughout the process. Therefore a system should be arrived at which is less likely to suffer from serious, large scale usability problems. Smaller scale interface issues can then be identified and dealt with by evaluation techniques.

There is no ‘right’ way of designing a system to be free of usability errors. The benefit of most of the techniques we have outlined in this unit are not that they tell a designer how to improve usability, but that they force the designer to think about the system from a ‘usability point of view’.

Below is a summary of the user centred design techniques we have discussed in the unit, stating in simple terms what they are good at and not good at. Once you have completed this course you should be more familiar with many of these techniques and should therefore be able to return to this table and add to and discuss it in more detail.

Technique	**Good For**	**Bad For**
Guidelines	Actually telling designers what they should or should not to achieve those goals.	Actually telling designers what they should or should not to achieve those goals.
Usability engineering	Actually telling designers what they should or should not to achieve those goals.	Actually telling designers what they should or should not to achieve those goals.
Rapid prototyping	Quickly testing designs with a user population.	Good software engineering – rapid prototyping can force design decisions to be made too early in the design process.
Design rationale	Recording why a system is designed the way it is.	Allowing developers to think that all designers options have been addressed.
Modelling techniques	Stripping away irrelevancies in a design so that the essentials can be easily analysed.	Allowing analysts to strip away essentials and analyse irrelevancies.
Evaluation techniques	Getting actual hard facts about what users do or think about systems.	Spotting big mistakes early enough that they can be cheaply remedied.

Extensions

Other development cycles

Obtain a good software engineering textbook (e.g. ‘Software Engineering’ by S. R. Schach, 1993, published by Irwin and Associates. A search for ‘software engineering’ on any technical library catalogue should bring up a wealth of good text books.) and investigate some of the design processes other than the waterfall model. Discuss the implications of user centred design for those process. Do those processes make user centred design easier or not? Why?

In particular you might wish to investigate the ‘Delta model’ which is held to be particularly adept at incorporating user centred design. What makes this so?

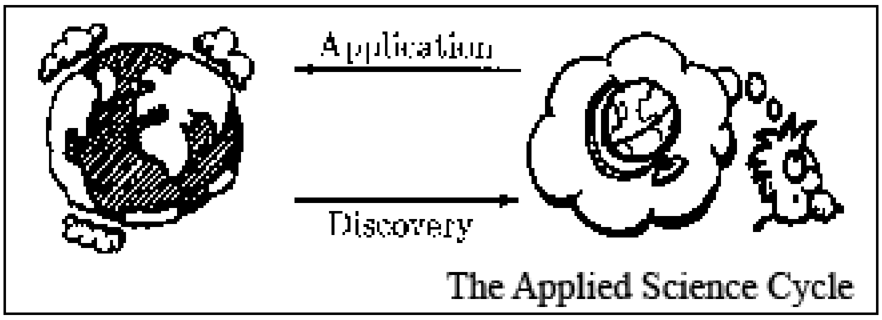
Is usability a science or craft skill?

Newton predicted that if you dropped a hammer and a feather in a vacuum they would fall at the same rate. This is because gravity pulls on objects consistently, independent of their mass. The only reason that hammers and feathers fall at different rates is because of friction with the air. Remove the air and they fall at the same rate. Newton was able to make this prediction based on a thorough and correct (not withstanding Einstein and relativity) understanding of the science of gravity. It was not until some hundreds of year later that an Apollo astronaut actually took a feather and hammer and dropped them on the moon that we actually had evidence that Newton was correct.

There is an equivalent question for HCI; do we understand users and user behaviour enough that we can predict how they will behave with an interactive system? If we do, then we can design for usability because we can predict user behaviour, but if we cannot then the best we can do is build a system, give it to users and see what they do with it.

The applied science cycle

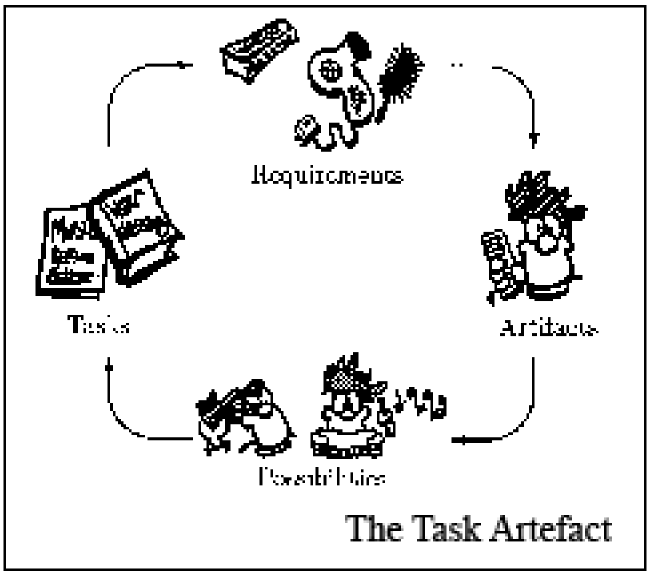
Barnard (1991) suggests the ‘Applied Science Cycle’ shown in figure 1. There is a ‘real world’ of tangible objects and observable behaviours and an ‘abstract world’ of scientific theories about the sort of things that go on in the real world. A scientific theory is held to be a good if it manages to accurately predict what goes on in the real world. (Newton's theories of gravity are good, even though they have been supplanted by Einstein's theories. Newton is accurate when describing gravity as experienced by those who are not approaching the speed of light. As most of us rarely approach the speed of light then Newton's theories are good for us.)



Barnard argues that things go on in the real world and that scientists observe them and think up theories to account for them. This translation from the real world to the abstract is called 'discovery'. Designers can use the theories in the abstract world to design artefacts in the real world. This translation from the abstract world to the real world is called 'application'.

Applying Barnard's cycle to HCI, we have a world view whereby people who study human behaviour (psychologists and sociologists) observe and analyse human behaviour and develop theories about their behaviour (discovery). Interactive system designers can then apply these theories to help them better design their systems (application).

The task artefact cycle



Carroll et al (1991) rejects Barnard's cycle on the grounds that is inappropriate for designing interactive systems. He argues that human behaviour is simply too complex to usefully abstract over and therefore the discovery translation of the applied science cycle is worthless.

Carroll suggests a replacement for the applied science cycle called the 'Task-artefact cycle' shown in figure 2. It is rather more subtle than Barnard's cycle so we have shown it with an example.

We start with artefacts; tools for doing a job. Here the tool we consider is a comb. A tool offers possibilities, and those possibilities may not be the ones for which it was originally designed. In the case of the comb it may be used for combing hair, or it may be wrapped in paper and played as a musical instrument. This gives us new tasks. From these tasks we can gather requirements from users about how they may wish to do their tasks better, in the case of the comb the user may require a harmonica if they wish to concentrate on their musical skills or better hair dressing tools if they stick to the comb's original purpose.

Central to Carroll's cycle is the notion of artefacts, whereas central to Barnard's cycle is abstract theories. Carroll therefore argues that his cycle is more practical and better reflects design practice in the real world. However Carroll's argument is that you must produce something and set it in the real world before you will be able to usefully understand how it is used. Taken to its logical conclusion Carroll's argument is for throwing mud against walls.

Barnard's argument should not really interpreted as a description of how things are, but how they should be. Useful and accurate theories of human computer interaction mean that systems can be designed based on those theories, by anyone who understands them.

The counter argument to theory based HCI is that of 'craft skill'. A craft skill is something that cannot be caught in a theory. Most artistic endeavours rely on craft skill. (Actually most scientists think that artistic endeavours rely on craft skill – when analysed most artists have a plethora of theories and rules which determine what they do.) Many HCI practitioners work as 'craft skill' experts; they will inform designers about what makes a good or bad interactive system, but the rationale behind their arguments is unclear and ill-formed. They have knowledge about HCI, but not in a form that is expressible. Hence they do not advance HCI as a science.

Carroll's arguments, particularly the argument that human behaviour cannot be abstracted over, lean towards HCI as a craft skill. It is not clear how a designer goes from 'requirements' in his cycle to 'artefacts' other than by applying craft skill.

Psychology and sociology as useful theories for HCI?

There is a well researched theory known as Fitt's Law (1954), developed by a psychologist. It describes very accurately the time it takes for the hand to move a certain distance and hit a target area. This law is very useful for designing buttons in graphical user interfaces (see Dix et al, page 252). It allows the designer to analyse the size and spacing of buttons and predict how easily the user (with the mouse) can press them.

Fitt's Law is a classic example of a discovery on Barnard's cycle which can be usefully applied. There are other examples of psychological theory which can be used to predict user behaviour. May, Scott and Barnard (1996) have developed theories of perception which predict how well users will be able to pick out icons on screen.

These 'small granularity' theories abound in psychology, describing hand to eye co-ordination, perception, our ability to think about two things at once, our ability to reason logically, etc.

Sociology looks at human behaviour at a much larger 'granularity', describing human group behaviour, particularly of interest to HCI is 'work theory' which describes how we go about doing our jobs and fulfilling tasks.

HCI sits between psychology and sociology. Interactive systems are seen as tools in work, so HCI is smaller scale than most sociology theories, but larger than most theories propounded in psychology. There are theories of HCI but they are not as well researched and established as those drawn from psychology and sociology. Also they are not clearly or abstractly set out as theories (at least not as abstractly set out as Fitt's Law, which is one mathematical equation) or as general as could be hoped.

Summary

In this extension unit we have dealt with some fairly lofty ideas about scientific validity and the use of science in design. The aim of the extension is to argue that we can design for usability, rather than just test for usability. We believe that it is possible and desirable to design for usability, although this approach is contentious.

Answers and Discussions

Answer to Review Question 1

Design techniques are applied during the design process in order to try and produce a system that is more usable. Summative evaluation techniques are applied when most of the design has been done and a tangible system has been produced. Evaluation is performed by giving a system to users and studying how they use it. Therefore evaluation can only take place when there is a tangible system to study. While design is taking place there will not be such a tangible system.

Answer to Review Question 2

During the early stages of the waterfall design process nothing tangible is actually produced. The early phases are about having and organising ideas about how to go about solving problems. These ideas exist on bits of paper or completely abstractly in the designers’ heads. They are therefore quite cheap to change if a mistake is found.

Later in the process actual tangible products are generated. These are much more expensive to change.

Answer to Review Question 3

A correct design step selects an option from the design space that solves the required problem. A good design step selects an option from the design space that solves the required problem well (e.g. efficiently, cheaply, etc). A good design step is by necessity correct, but not necessarily vice versa.

Answer to Review Question 4

Evolutionary development is best described by the task artefact cycle. It describes how the use of objects can change through time, and how the design of those objects needs to be updated to match those changed uses. This change of needs, use and objects through time is what evolutionary development is all about.

Answer to Review Question 5

Rapid prototyping actually involves users in the design, whereas the other techniques for user centred design rely on more abstract theories of user behaviour.

Answer to Review Question 6

Based on the analysis given the first option is probably the best. Screen real estate is the largest consideration to be made and therefore the second option is ruled out on those grounds. The third option is appealing, but even so it would still possible for the user to get lost in a deep menu hierarchy. The first option may limit the telephone functionality, but 258 functions should be far more than sufficient for a mobile phone. If more than 258 functions are required, this should raise much more fundamental questions about the design of the phone. The fact that the second option gives the user perfect knowledge as to where they are can be discounted to a great extent, because giving the user perfect knowledge is probably unnecessary, and that giving the user hints is probably sufficient.

However the option space we have described in the example is small. In particular there is no reason why the designer cannot implement combinations of the options. In this light a combination of options one and three gives a very good solution as it would have no seriously negative criteria.

Answer to Review Question 7

Because of the size of the user population devising a questionnaire would be the best solution. As well as designing the questionnaire well you need to consider how you are going to induce users to answer it. If you are delivering the product over the web then you could make answering the questionnaire a preliminary to downloading the product (but beware, this means the user will not have actually used the product). You may instead wish to offer an incentive like free support if the user completes the questionnaire.

Answer to Review Question 8

Mainly because users do not read manuals. This is not the users' fault; they have much better things to do with their time than trawl through manuals, none of which make very interesting reading. A well designed, usable consumer product should not need a manual.

Discussion point

Discuss the process you went through in this unit and unit 2 to analyse and redesign a web browser. What have you learnt? In particular we wanted to show that designing for usability need not be very difficult. In subsequent units you will be introduced to some complicated techniques that may daunt you.

Consider the amount of effort you have put into this analysis: it should have taken you about three hours. But with just a few hours of gathering evidence and careful critical thinking we have come up with a redesigned web browser that we claim is better suited to what we do with the web. We have employed no special skills other than critical thinking. Microsoft spend a huge amount of money per year on usability testing, but we have suggested improvements to one of their most visible products in a few, cheap hours. Why is that? Who is in the wrong? Us or Microsoft? Why do Microsoft spend this amount of money when it apparently does not greatly improve their products?

Discussion of Activity 1

Returning to the mobile phone menu problem we discussed in the usability engineering section, a question may be 'How to prevent users becoming lost in menus?'

Options for this question could be:

1. to flatten menu hierarchies to a maximum depth of three,
2. to give textual feedback as to the current menu display (e.g. display the text 'Address book : Edit : Add number' to show that the user has selected the 'Address book' option from the main menu, the 'Edit' option from the Address book menu, and 'Add number' option from the edit menu.
3. to give graphical feedback as to how deep the user is in the menu hierarchy by shifting each new menu selected a little down and to the left, giving the impression of stacking up menus.

For each of these options there are several criteria effecting whether or not it is a good design decision:

1. Screen real estate
2. Limited functionality
3. Accurate user knowledge as to where they are in the hierarchy

Now we can go through discussing each of these criteria for each of the options.

1. If menu depth is kept to a maximum of three then:
   - this has no real effect on screen real estate,

- it limits the functions that can be placed in the phone. Say a maximum of six items per menu can be displayed, then there is a maximum of 258 functions that the phone can perform,
- keeping the menu hierarchy flat, improves the chances of the user remembering where they are.

2. If textual feedback is given then:
   - there will be problems with screen real estate, textual descriptions take up a lot of space,
   - there is no limit to the number of functions,
   - user knowledge of where they are will be very accurate.
3. If graphical feed back is given then:
   - screen real estate can be efficiently used,
   - there is no limit to the number of functions,
   - the user will have a prompt as to where they are in the hierarchy, but not a very accurate one.

So for the question we have identified three options and three criteria which effect those options.

Discussion of Activity 2

The answer is to where the browser finishes up after press Back twice is: ‘it depends’. If you moved from page to D by following links then pressing the back button twice will put you on page C. If you went from D to B by pressing the Back button twice then pressing the Back button twice on page H will send you to page A.

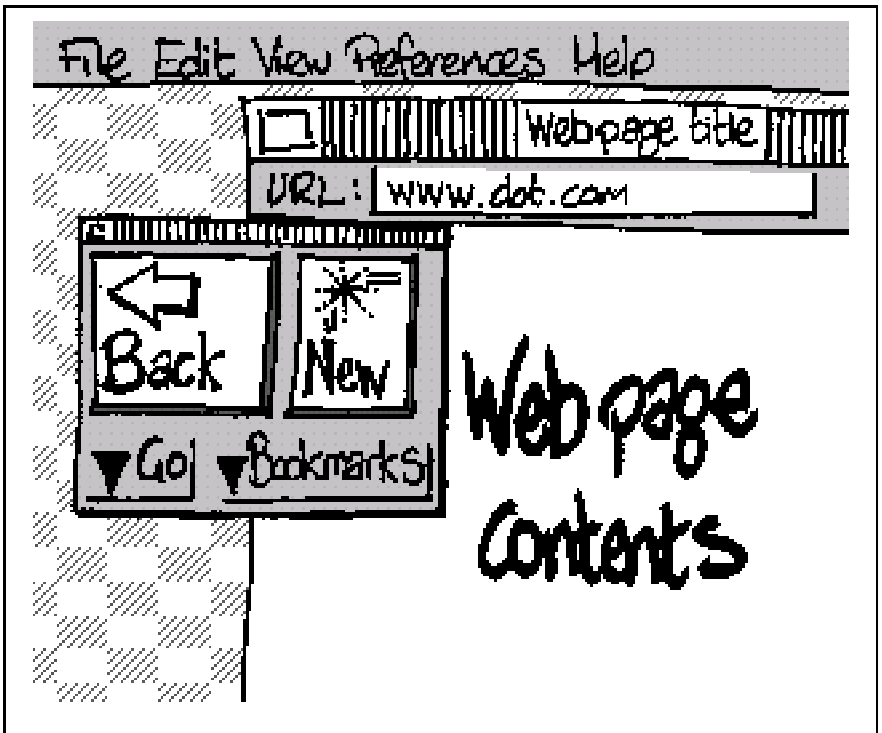
It turns out that what the Back button does is simple: it moves you back one place in the list of pages on the Go menu, but what is actually on the Go menu is not easy to work out, and therefore what the Back actually does is not very predictable. If you are on the top page on the Go menu then jumping along a hyperlink adds the new page you jump to on to the top of the Go menu. If, however, you jump along a hyperlink when not at the top of the Go menu then the pages above the current one are deleted from the Go menu and replaced by the page jumped to. The Go menu is therefore no a complete history of where the browser has been. Many users are not aware of this.

Discussion of Activity 3

Think about a menu system on a normal computer with a full sized screen. It is difficult for the user to get lost within that menu system, because the computer gives very explicit feedback as to where the user is in the menu hierarchy. On a restricted size screen there is no room to give the user such obvious feedback as to where they are, so you must consider (at least) two things: ways of giving the user feedback as to where they are which uses very little screen real estate, and ways of making the menu hierarchy simple enough that the user does not get lost.

Discussion of Activity 4

A sketch of my interface is shown below.I have made the navigation controls into a floating window and made the back button largest in that window. The Go menu now drops down from the navigation window and the bookmarks can also be shown in a floating window. The URL of the current page is shown across the top of the web page window and can be edited. All other functionality is in the menus. I have moved the ‘New navigator’ function out of the menus and onto the navigation window. I use multiple windows a lot and find it annoying having to rattle about in the menus to create a new browser. Also, instead of starting a new browser with the predefined ‘home page’ it creates a copy of the current page. This is so that when I find a page with several links on it that are interesting and I may want to come back to I can easily make a copy of that page and come back to it later to investigate the other links.



Sketch of a redesigned web interface

I have sketched out this interface to support my use of the web, your use may be different, and therefore you will have hopefully come up with a different design. So much the better.

Discussion of Activity 5

If Mozilla Firefox and Microsoft Internet Explorer show feature accretion then that is because Mozilla and Microsoft consider that loading their applications up with as many features as possible is an advantage. Putting many features in a product can give that product the appearance of being ‘powerful’ or ‘professional’. An argument that those features are useless, and worse, get in the way, is much less tangible and difficult to demonstrate, and therefore does not become much of a negative point.

Chapter 4. Cognitive Psychology

Table of Contents

Context

To carry out effective User Centred Design, designers need to something of how humans process information: how they perceive the world, focus their attention, use memory, make choices and respond. Cognitive psychology is the study of such information processing. Knowing the capabilities and limitations of your users will lead to you making better design choices. This Unit introduces cognitive psychology and deals with one key aspect, namely visual perception – i.e., how we process visual information that is presented to us. You will learn about theories of visual perception and how this knowledge might help build better interactive systems.

Objectives

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

- Define cognitive psychology
- Give reasons why interactive system designers should study cognitive psychology
- Discuss 2 theories of visual perception. The ones presented are constructivism and the ecological approach.
- Show how knowledge of the two theories can improve in, for example, icon (constructivism) and affordance design (ecological approach)

Cognitive Psychology

Cognitive psychology is the branch of psychology concerned with how our minds process the information sent from our various senses. Most of the work from cognitive psychologists has been to quantify how efficient we are at processing this type of information – how much we can process and how quickly. More recently, however, cognitive psychologists have become interested in how the mind processes information generated from working with other humans and external artefacts such as computers.

The study of cognitive psychology as a branch of psychology was started around the same time as researchers became interested in the possibility of artificial intelligence (roughly in the mid 1950's). Psychologists believed that the human brain was some sort of information processing machine, which was fed input from senses and stored images, thoughts etc. in memory. The discipline of cognitive psychology describes the mind as:

" *A general purpose system for processing symbols that is limited by both structural and resource limitations.*"

In this context, a symbol is a pattern stored in memory which represents, or 'points-to' something in the external world. Various processes in the mind manipulate and transform symbols from one sort to another. The goal of cognitive psychologists is to define these processes and representations; to give an understanding of how well our mind will perform a given task.

Obviously, human–computer interaction can benefit from this work by building models of user performance and defining design guidelines which are sympathetic to the way our mind processes the information being displayed on a computer screen. Cognitive psychology also offers the possibility of building user models – essentially computer programs which react the same way to certain stimuli as humans. These models can be used to test an interface and predict the types of problems users could experience. Cognitive psychology can also be used after an interface has been constructed to give insight into why users are experiencing problems with that interface.

In a future unit, you will be shown different models of how the brain processes its inputs. In this unit, however, we are primarily concerned with how humans process visual information. We will investigate how our minds process visual information and how this processing can be exploited to improve interface design. You will also discover that our eye is a lot more complex than a simple camera, and that our minds often "embellish" the images received from the eye.

Review Question 1

List the main goals of cognitive psychologists

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

Review Question2

What can cognitive psychology offer computer interface designers?

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

Visual Perception

Have you ever stopped to consider how amazing the human vision system is? It allows you to see in very bright and dull environments; it allows you to detect minute variations in colour and it also allows you to detect rapid movement, be that a darting animal or a sudden flash of lightning. However, the eye is not perfect. We are limited to a narrow spectrum of colours (between ultra-violet and infra-red). Some objects move too quickly (or too slowly) to see any movement.

These aspects, however, are governed by the physical composition of the eye. What we are interested in is how the brain processes the information which the eye does manage to capture. If we are to create usable interfaces, then we need to examine how the brain processes the information presented to it.

There are a number of theories which have been proposed to explain how we process what we see. Broadly speaking, they fall in to two main categories: constructivist and ecological. Constructivist theories state that our visual perception of the world is constructed by applying our knowledge of the world to what our eyes are currently seeing. Ecological theories, however, state that there is no high level processing, we extract information directly from the light captured by the eye. Neither approach is exclusively correct and there is much evidence supporting both points of view.

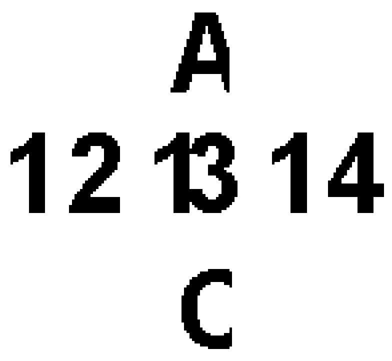
Constructivism

Constructivist theories are all based on the idea that we do not perceive the world directly – that we process any image we see before it is consciously recognised. (There are many different types of constructivist theory, perhaps the best known is Gestalt theory which we shall discuss in the next section.) This has quite a few, far reaching, implications For instance, one consequence of the constructivist approach is that a child must learn some base amount of knowledge before it can ‘see’ (as we understand the term ‘see’).

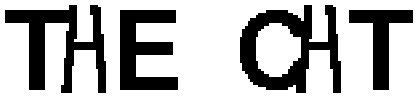
To illustrate their point, the constructivists show how the mind uses contextual information and world knowledge to affect what we see. By using optical illusions we are challenged to question if what our eye detects is really what we “see.”

Theory - Contextual Information

Consider the following figure



What do you see? Is the middle symbol a “B” or a “13” ? The context in which we see a symbol has a huge influence over how we “see” it. What about the sentence in figure2? The symbol in the middle of each word is clearly identical, yet we are able to “see” it differently in each context!



Our brain is able to use information about the context in which a symbol appears. It can use the contextual information to disambiguate the symbol, so that we “see” complete and meaningful words.

Application - Contextual information

Contextual perception obviously has applications in interface design. For instance an icon design which means one thing in one part of the interface, might mean something completely different in another place. A good example of this is the check box.

Usually this symbol is used to show whether an item is selected or not. But, it is also the symbol used to denote “Close” when used in the top right corner of a window in Windows 95/98/2000. Again, in the different contexts, we see different buttons, even though the representation is identical.

Activity 1 - Using Context Information

Can you find examples of interface elements (like the check box) which appear identical, but have different meanings in different contexts? You may find that pictures denoting actions/verbs rely more on context than those representing nouns.

A discussion on this activity can be found at the end of the chapter.

Review Question 3

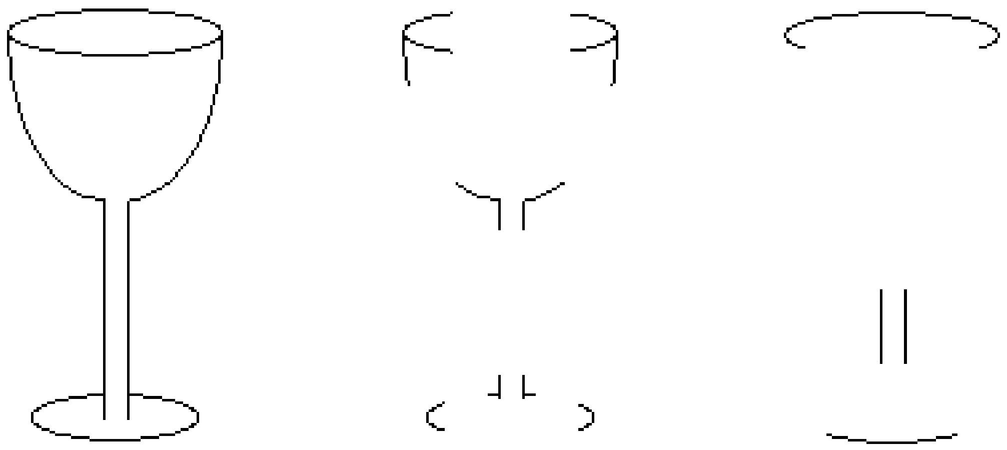
Explain why context is important when designing an interface?

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

Theory - World Knowledge

From childhood, humans develop a huge knowledge about objects and how to recognise them. Our perceptual systems exploit this knowledge, allowing us to recognise objects when parts of that object are obscured. In figure 3 below, we can readily identify the wine glass, even though roughly half the original information is obscured. The final glass is much harder to identify. Not only is much more information missing, but crucial details (such as the contour of the glass) is not present.

Using world knowledge to recognise partial shapes



A lot of research has been conducted in to the types of information our perceptual systems need to identify objects. Edge information and contour information is vital – other information, in particular colour, is largely redundant. (Even an object which is heavily reliant on colour to identify it, such as a banana, can be as quickly recognised in black and white).

Application : World Knowledge

Again, this information about our ability to perceive incomplete objects is useful at the user interface. When designing graphics were screen area is restricted (as is the case for an icon), we can show those edges which most define an object, without worrying about colour information.

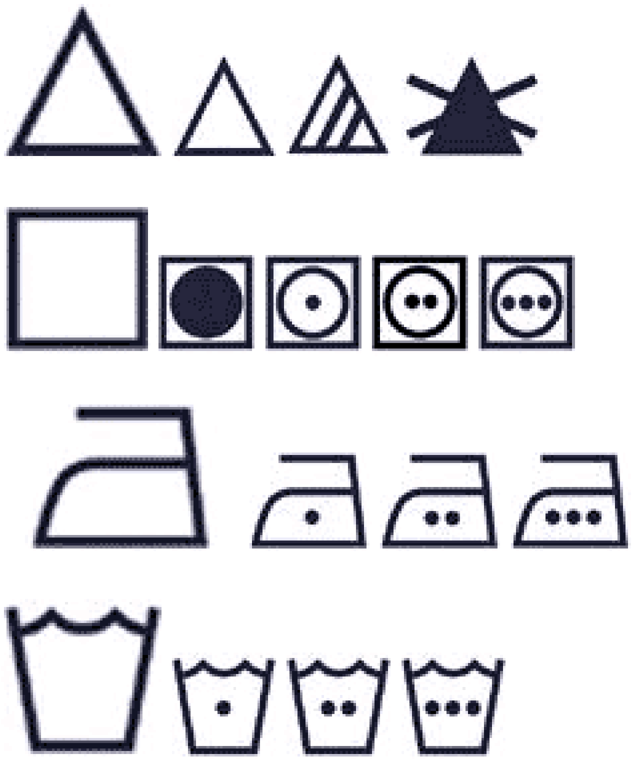
Activity 2– Constructivism; using world knowledge

Look at the icons at the top of your browser window (Back, Forward etc.) and have a go at re-designing them to exploit your world knowledge.

A discussion on this activity can be found at the end of the chapter.

Activity 3 – Icon design

Look at the laundry labels in the images beneath. Each represents a separate class of operation you can perform on your laundry (e.g. dry, wash, dry clean and bleach). Can you guess which symbol represents each operation? Do you think that the symbols exploit world knowledge sufficiently well to depict each action? If not, suggest some alternatives.

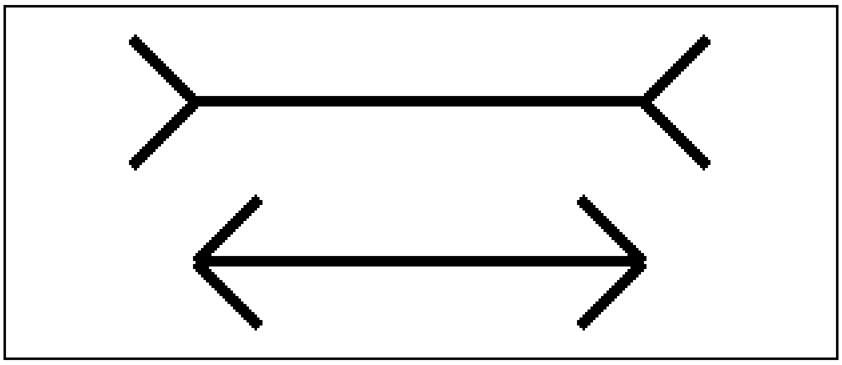


A discussion on this activity can be found at the end of the chapter.

Gestalt psychology

The term gestalt refers to something which has a well structured form; a form which is often more than the sum of its parts. For instance, four lines arranged in a square form something which we primarily recognise as ‘square’ – not as a collection of four lines arranged in some fashion. Gestalt theory would argue that our perception of the lines is influenced by the knowledge we have about geometric shapes (in this case the square) which causes us to perceive a square and not individual lines. The constructivism described by Gestalt theory, therefore, is the way our mind’s knowledge of different types of form affect our perception.

The relationship between the lines in the Müller-Lyer illusion (below) serves as a good example of how our brain constructs something more from the lines than is actually found in the diagram. In this instance, our brain constructs views from the two groups of three lines such that the top horizontal line looks longer than the horizontal line beneath it. In actual fact, there are both the same length.



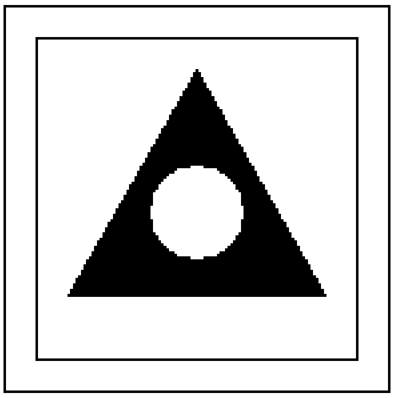
Review Question 4

What is meant by a 'gestalt,' illustrating your answer with an example?

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

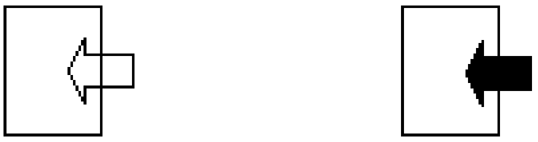
Theory : Figure and Ground

One key “form” in the Gestalt theory is the idea of “Figure and Ground.” Look at the figure below. What do you see? Do you see a complete triangle with a circle placed on top of it? Or, do you see a triangle with a circular hole through which you can see the square beneath? In other words, is the circle part of the ‘figure’ or is it part of the ‘ground’?



Presenting the distinction between figure and ground is essential if we are to produce pictures and icons which are unambiguous. Consider the following symbols (figure 6) which could be used to represent “exit”:

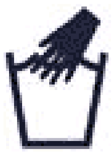
Application : Figure and Ground



When drawing images like this on a page, or a computer screen, you need to be careful that you are not confusing your user about what is part of the foreground and what is part of the background in your image. The idea of “figure and ground” has implications when designing symbols such as icons. One rule we can infer is in marking boundaries: it is better to use a solid shape than just[1] an outline. By using a contrast boundary, the black arrow stands out more than the outlined arrow.

Activity 4

Consider the following laundry symbol. Could it be improved using figure and ground rules?

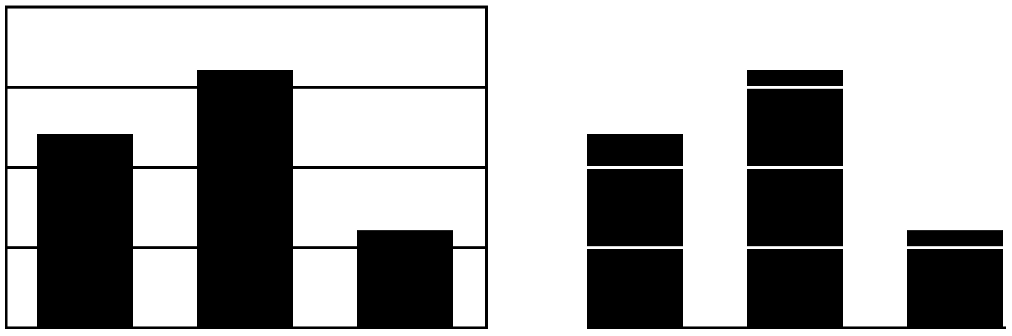


A discussion on this activity can be found at the end of the chapter.

Too much foreground

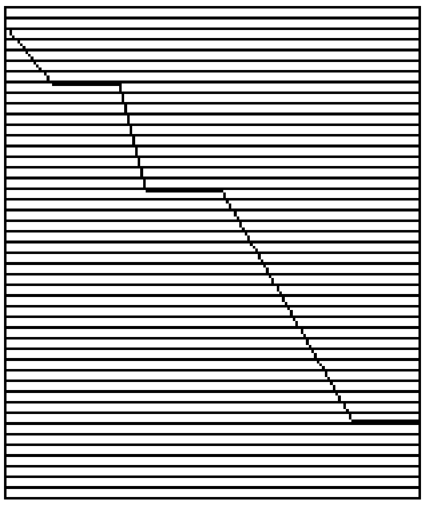
Another application of figure and ground distinction is increasing the amount of information conveyed without increasing the amount of “ink” used to represent that information. Often, computer graphics can become cluttered and hard to read because the designer is trying to convey a lot of information in a small space.

To illustrate the point, consider the graph on the left. Here the information being presented (as represented by the height of the bars) is drawn in the same colour as the contextual information (the lines of the graph). The graph on the right, however, exploits figure and ground to show the information in one colour but keep the context information as part of the ground.



Activity 5 – Foreground clutter

How can we improve the following graph using the rules of figure and ground?

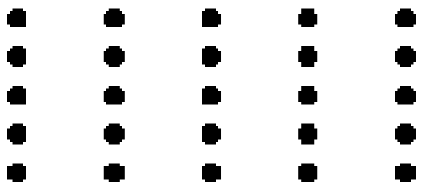


A discussion on this activity can be found at the end of the chapter.

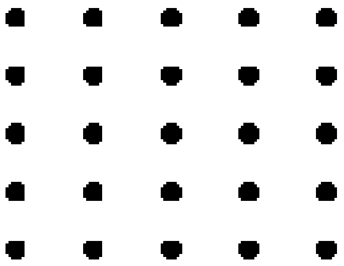
Theory : Perceptual Organisation and grouping laws

One of the main contributions of Gestalt theory is in the principles by which objects group themselves in visual perception. There are five different types of grouping:

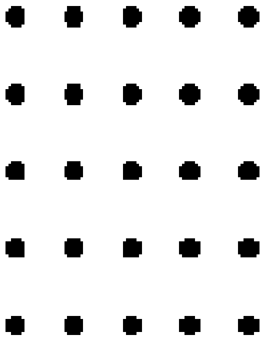
Proximity: In the figure below, the dots could be organised in rows or columns.



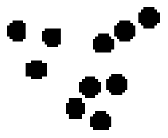
Due to the proximity of dots in the figure below, the dots arrange themselves in columns.



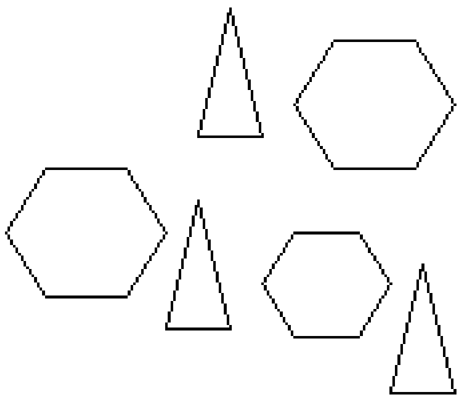
Conversely, the dots in the figure below appear in rows.



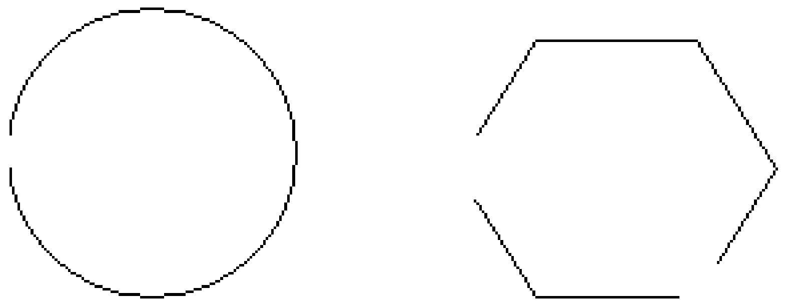
Proximity need not only apply to organised items, then tendency to group by proximity can be seen in random allocations such as in the figure below:



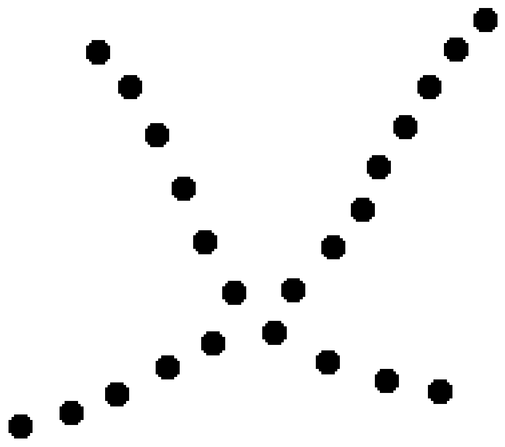
Similarity: Objects of similar shape or colour will be perceived to be grouped together, as in figure below.



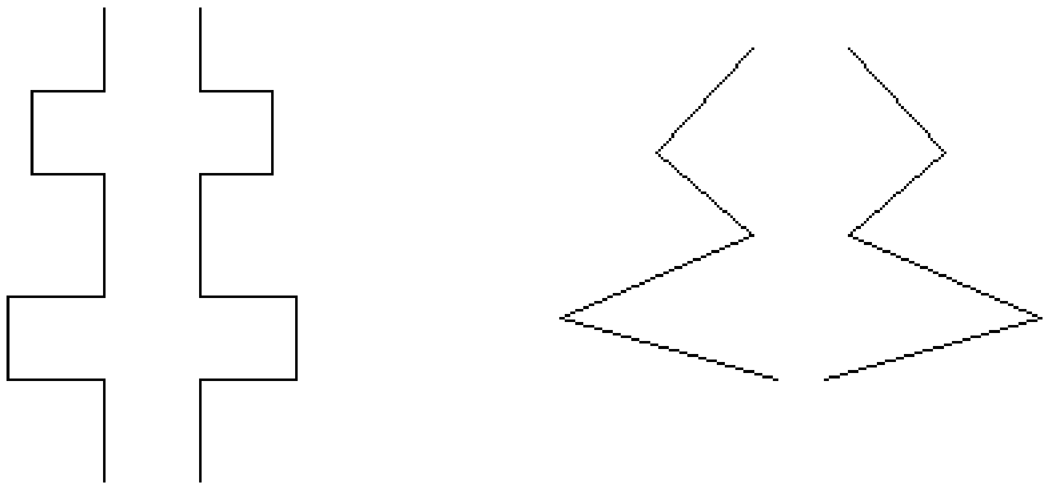
Closure: Rather than appearing as three separate lines, the application of closure to perception means that we see a circle and hexagon in this figure below:



Continuity: By using continuity in perception, we tend to see two distinct trails of dots in the following figure.



Symmetry: Areas that are surrounded by symmetrical lines tend to be recognised as shapes, rather than the lines being perceived as shapes in their own right. (see below)

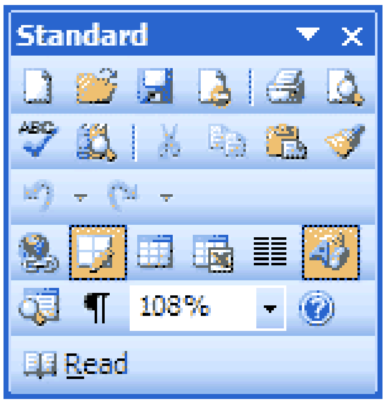


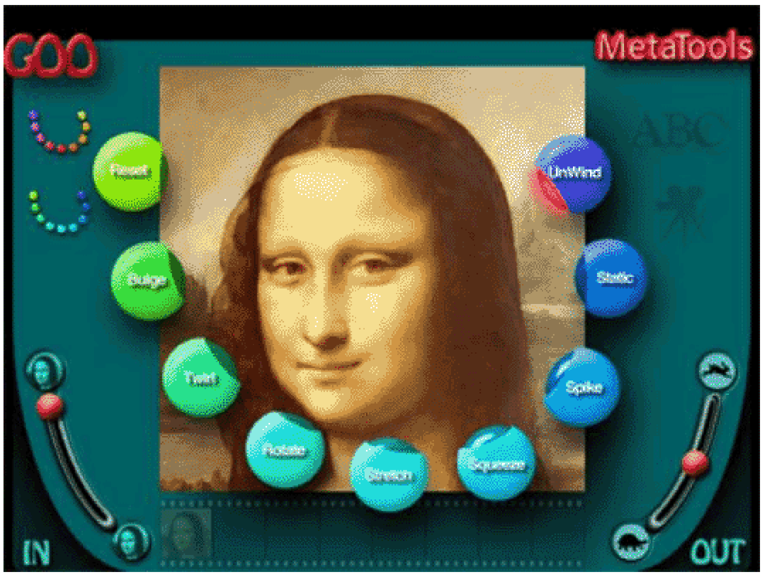
Application : Perceptual Organisation and grouping laws

Activity 6

The five grouping rules discussed above give the interface designers ideas about how to arrange interface objects so that they will be perceived to belong together in some way.

Consider the following dialog boxes and tool bars from various applications. Discuss how grouping rules have been used to design each one.

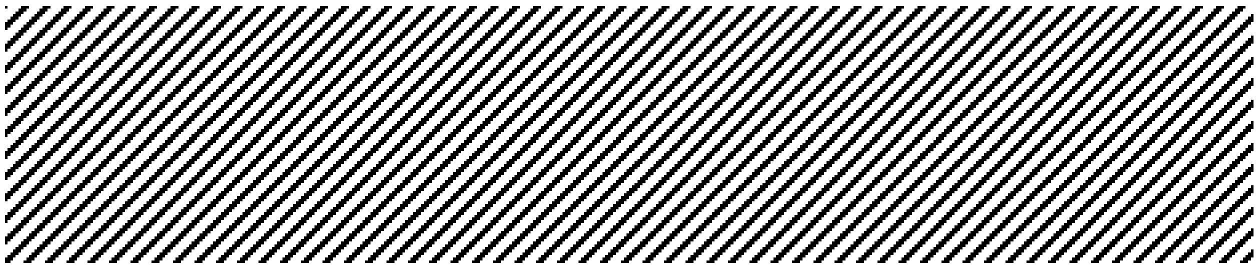




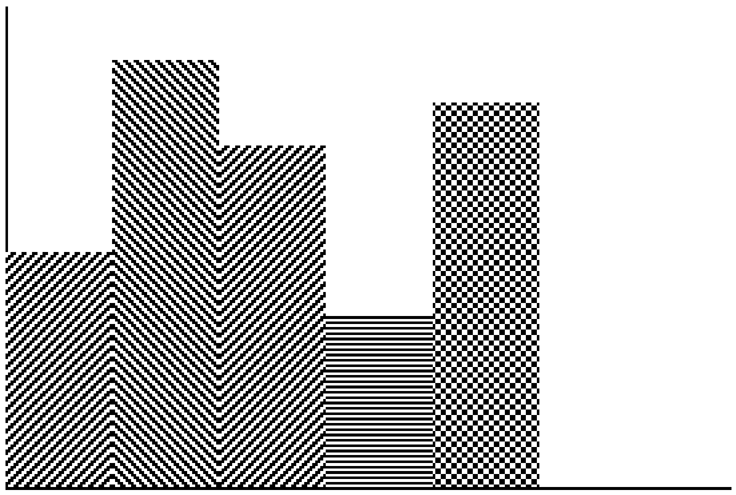
A discussion on this activity can be found at the end of the chapter.

Grouping too closely

Grouping objects too closely together can play tricks on your perceptual system, causing an illusion called moiré movement. Stare at the following image and you should experience the effect.



This can be a problem in computer graphics, as in figure below,



Review Question 5

Choose the most appropriate word from *Symmetry*, *Similarity*, *Closure*, *Continuity* or *Proximity* to describe each of the Gestalt grouping descriptions beneath.

- Objects arranged regularly, but which are close vertically, tend to be grouped in rows
- Objects of similar shape or colour will be perceived to be grouped together
- Objects which follow each other in an un-broken line are grouped together
- Even though parts of regular objects are missing, we tend to fill in the gaps

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

Ecologist

There is a lot of evidence to support the theories of constructivist perception. In particular, the grouping laws of Gestalt theory have proven themselves in a wide variety of experiments and applications. But do our minds process every image we see in such detail?

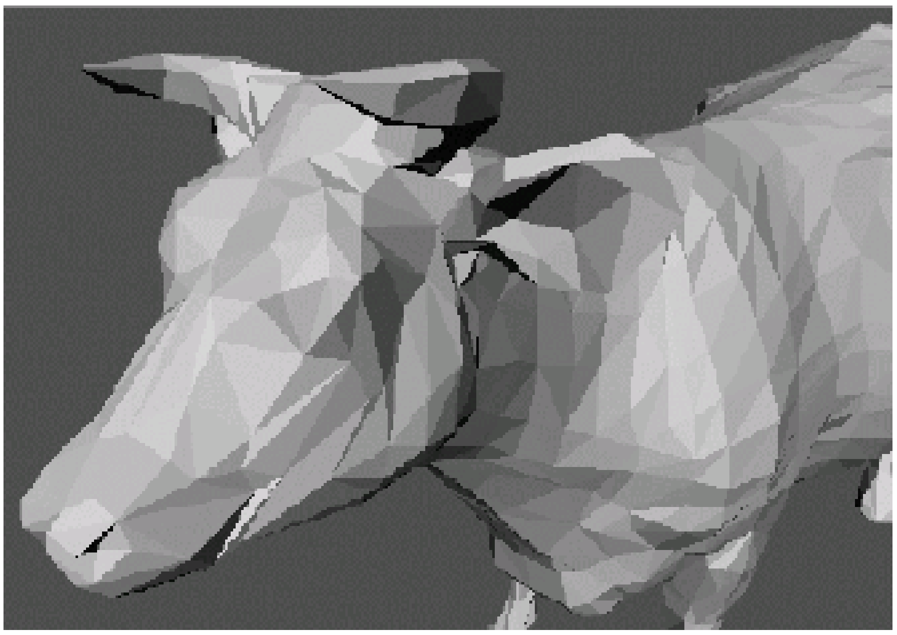
Whilst working on training fighter pilots in the second world war, psychologist J.J Gibson came to the conclusion that constructivism was not the whole story. Instead, Gibson and his followers claimed that we could perceive information directly from our environment without any higher level processing. Ecologists argue that our eyes and minds have adapted to our environment over millions of years of evolution and therefore are optimised to perceive that environment very efficiently. Furthermore, they argue that the experiments of the constructivists operate in a controlled environment (the laboratory) which cannot capture how an eye perceives in its natural environment. Ecologists are therefore interested in what cues our eyes can take directly from the light in the surrounding environment.

Theory - Optical flow and Invariants

One of Gibson’s main arguments about perception is that movement is a crucial part of the process. Our eyes are rarely presented with a static view of a scene, instead we tend to walk around, giving our eyes a constantly changing range of viewpoints. Consider a textured surface which we are trying to observe. As we walk closer the that surface, our perception of it changes are we are able to see increasing levels of detail. The changes in this surface are not random – there is a gentle change in detail to which our eyes are accustomed. Gibson calls this change in perception optic flow.

Application - Optical flow and Invariants

Reproducing optical flow convincingly is a big problem in computer interfaces. In a virtual environment, for instance, we can model object surfaces by giving them a fixed texture. As we walk around the virtual environment, objects change size in accordance with perspective rules (object sizes can be rescaled according to well understood mathematical rules.) What does not change, however, is the texture, or detailing, on surfaces. The transitions in surface detail are not well understood and quite often the “solution” is to store a model for remote viewing and one for close viewing. The resultant effect is unsettling as object suddenly change from blurred to pin-sharp detail with only the smallest of movements through the virtual environment. This problem is called popping and can be seen in the images below. In figure below, the viewer is standing back from the cow.



As they move a few millimetres forward, the computer decides that they should be able to see more detail and suddenly the cow has an eye, as can be seen in this figure.

Optical flow turns out to be just one case in a wider family of invariants. “Invariant” is the word Gibson uses for any perceived pattern of change within the observed environment – invariants are the patterns of change which are familiar to our visual system. One particularly important class of invariant is the affordance, which we shall investigate next.

Review Question 6

What is a moiré movement and what does it tell us about the visual system?

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

Theory : Affordance

One idea of Gibson’s which is crucial to interface design, is that of an affordance. Gibson says that our environment contains invariant information, the successful detection of which has survival implications for the observer. These affordances which objects provide give the observer clues about how they might be used – sit on-able, grasp-able, throw-able etc. In effect, the affordances provide the meaning the environment has for the observer, showing them what is possible within that environment. The interaction possibilities provided by the affordances has been called the effectivities of an environment.

Remember that the ecologists are claiming that these affordances are being perceived directly – there is no conscious processing. Therefore, the properties of an object which make it appear graspable can be directly inferred from nothing more than the reflected light from that object striking our eyes.

This is quite a bold claim, which the ecologists defend by arguing the subtle and inextricable links between observer and observed environment. Regardless of the exact mechanism of how we perceive affordances, there effect in computer interfaces is profound.

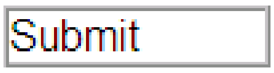
Review Question 7

What is popping and how does it relate to the idea of optic flow?

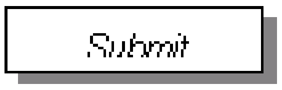
Answer to this question can be found at the end of the chapter.

Application : Affordances (Donald Norman)

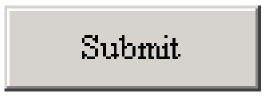
One psychologist who recognised the importance of affordances in designing interfaces was Donald Norman. He showed that interface elements could be designed to afford how the user should interact with them. Consider the button below, part of the original Macintosh interface.



How do we know that this is in fact a button? How does it afford pushing? A better version of the button would be…

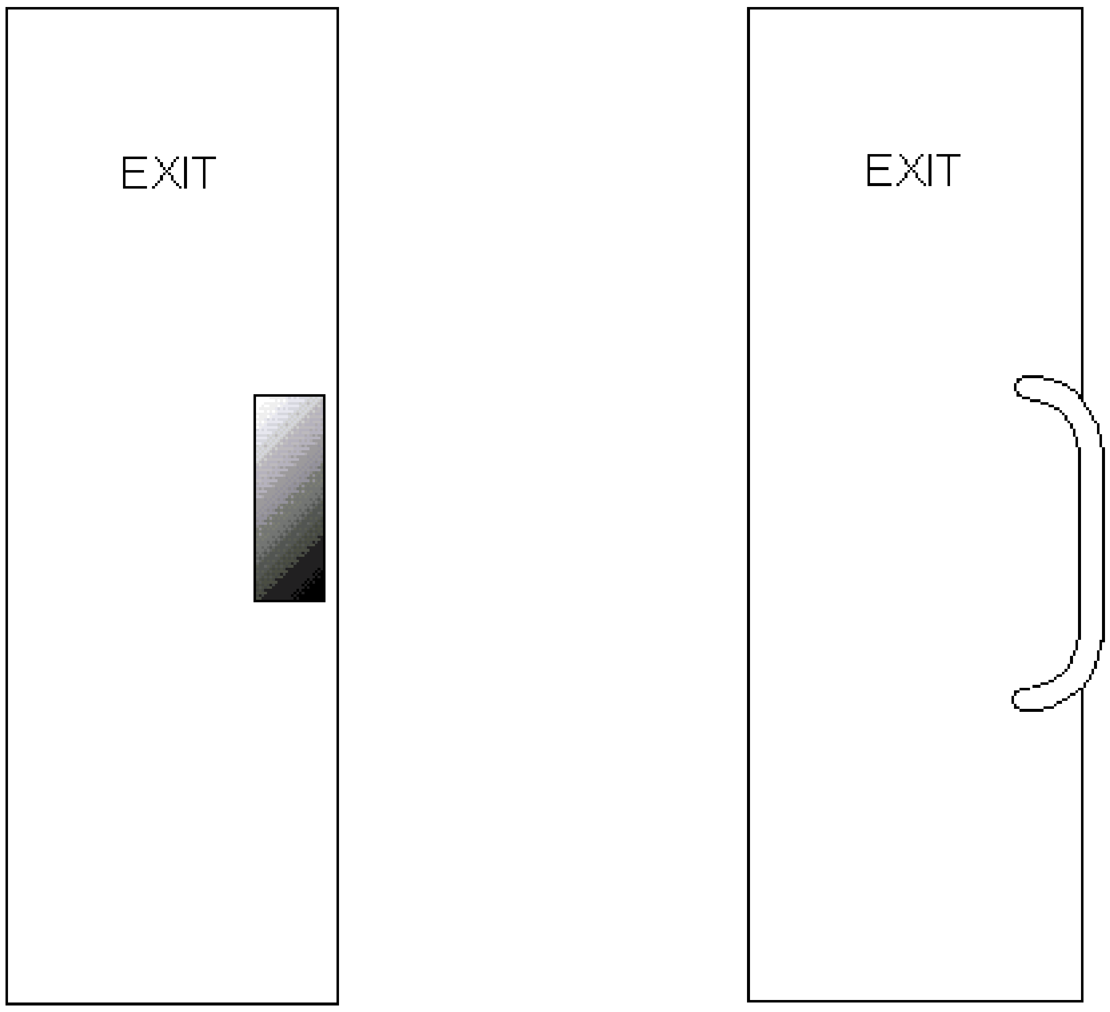


In this case, we can see that the button is raised off the page, so could perhaps be pushed level with the page. As computers became more powerful, colour could be added to interfaces providing even better affordances, as in…



By using a bevelled edge and a metallic colour, the button now looks like its real world counterpart. The result is an on screen button which provides the same affordances as its more familiar real-world counterpart.

Norman used a wide variety of examples ranging from computer interfaces to door handles showing how badly designed objects could afford one style of interaction, but actually use another. A common example of this are door handles. Handles of the type found in the left side afford pushing whilst those on the right afford pulling. Quite often the plates on the left door are used for pulling the door open. This is why people often spend ages pushing pull doors (or indeed pulling push doors) as the handle affords the wrong type of interaction. Where handles are used inappropriately, they are often marked with an instruction.



Review Question 8

What is the key difference between the ecologists and the constructivists?

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

Review Question 9

Why is this door not going to work for most humans?

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

Activity 7 - Web Affordance

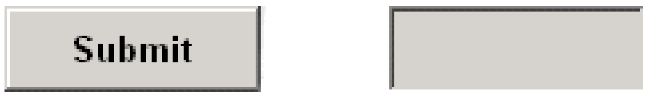
On a Web page, how do you know what is a hyperlink and what is regular text or image? What affordance is provided?

Take an example Web page with poor affordance and re-design it to improve its affordances.

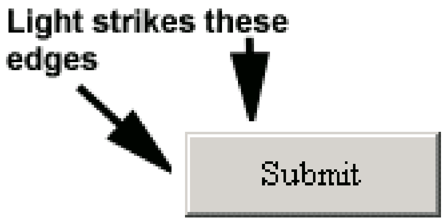
A discussion on this activity can be found at the end of the chapter.

Reinforcing Affordances - Shading Convention

Look at the two graphics below. One is a button and the other is a text box. How do you know which is which?



The one on the left is the button, the one on the right is the text box. The only difference between them is the shading on their edges. In the convention of user interfaces, buttons are raised and text boxes are recessed. This only works, however, if we imagine a light source shining from the top left hand corner of the screen.



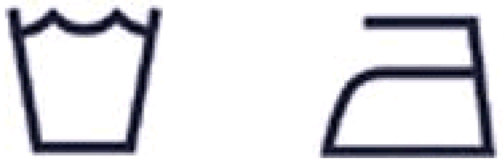
If the light shone from the bottom right of the screen, we would perceive the left image to be the text box, and the right hand image to be the button.

Fortunately all graphical user interfaces assume a light source at the top left hand corner of the screen. Therefore, any image or icon you create on the screen should have its top and left edge drawn in a lighter colour if it is to be perceived as a raised object. All recessed images should have their top and left edges drawn in a darker colour.

Look at the buttons at the top of this browser and you will see that the top and left edges are drawn in a lighter colour than the rest of the button.

Activity 8

Draw three-dimensional versions of the following symbols – make the first raised and the second recessed.



A discussion on this activity can be found at the end of the chapter.

Culture

One consideration to bear in mind is that our perceptual systems are shaped by the culture in which we grow up. Therefore, if you are creating an interface for a different culture than your own, it is well worth investigating cultural differences between you and the target user group. Culture can affect both constructivist and ecological perception, as seen in the following examples.

Affordances between cultures can also become confused. Work from anthropologists reveals many anecdotes such as the tribe which when first were presented with spoons used the bowl of the spoon to grasp and scooped with the handle.

So, if you are designing an interface for a culture other than your own, spend some time learning how their culture may have influenced their visual perception systems.

Activity 9

Although not strictly a perception problem, there are strong cultural reasons why the most used menu (usually the “File” menu) is in the top left corner of the screen. Why is this and how would you change the interface for a user who is from China?

A discussion on this activity can be found at the end of the chapter.

Discussion Topics

How do you think we see? Do you think that the constructivist and ecological approaches fully describe all the aspects of the human perceptual system? Think about the following issues:

Can you think of any visual effect, which is not adequately described by the two theories we have looked at? What are the biggest strengths and weaknesses of each theory? Do you think that the theories are mutually exclusive, or can you use on theory in some instances and the other theory at other times? What do you think about affordances – it is really an ecological phenomenon, or is there some constructivism at play? What do you think of the constructivist's assertion that babies see in a completely different way to adults?

Answers and Discussions

Answer to Review Question 1

To understand what processes our brain uses to understand and store the input (symbols) it receives.

Answer to Review Question 2

By understanding how our brain processes information, we can design interfaces, which exploit the strengths of how our brain works.

Answer to Review Question 3

Context gives meaning to a symbol. Without context, symbols such as the box with a cross in it can be ambiguous. Context can remove that ambiguity.

Answer to Review Question 4

A gestalt is a recognisable form; a standard object which our perceptual system can readily identify. Examples range from geometric shapes, through to more complex common structures, like wineglasses.

Answer to Review Question 5

Proximity, Similarity, Continuity, Closure

Answer to Review Question 6

Moire effects arrive from the perception of movement, a shimmering, in lines that are grouped too closely together. Essentially it tells us that we cannot trust our visual perception – patently the page is not moving but yet we see movement.

Answer to Review Question 7

Popping is the sudden appearing and disappearing of detail as you move around in virtual reality. This is at odds with optic flow, which says that detail changes in a smooth way, as our eyes are able to detect gradually more and less information.

Answer to Review Question 8

Constructivists say that we only perceive images once our brain has worked on them. Ecologists say that a scene is perceived as soon as the light from that scene is received by our eyes.

Answer to Review Question 9

Ecologists tell us that affordances are acted on without processing by the brain. Therefore humans will attempt to push this door before our brain has a chance to process the word “Pull” on the handle and tell our conscious selves to pull instead of push.

Discussion on Activity 1

One group of symbols which have different meanings are arrows. For instance the back arrow can mean “scroll left” or “Undo” or “Go back to previous Web page” depending on context. Actions tend to be more ambiguous than nouns as nouns refer to concrete static objects whilst actions describe transformations to nouns which happen over time – this could be solved by providing animated icons.

Discussion on Activity 2

No answer is possible to provide here as it relies entirely on how you perceive the Web.

Discussion on Activity 3

The triangles represent bleach. The crossed out triangle obviously means no bleach.

The squares are the tumble dryers. The dots in the centre of the squares represent the heat at which the garment may be dried – three dots being the hottest.

The third row of symbols represents ironing. Again, the dots represent the heat of the iron.

Finally, the last row represents how the garment should be washed. Dots denote temperature and a hand in the bucket would indicate hand wash only.

These symbols would be more effective if laundry equipment (e.g. irons and washing machines) used the same symbols so that the user need not even understand the meaning of the symbol. This is the best solution as you will discover when you try to design alternatives that they will most likely only make sense to you. If you sell garments worldwide, symbols will inevitably be miss-interpreted.

Discussion on Activity 4

Possibly. It may be possible to better separate the water line from the hand line. However, our perceptual system, using the gestalt laws allows us to group the water and the hand separately, so they are not confused.

Discussion on Activity 5

Simply remove the horizontal lines (apart from the X-axis) and the right-most vertical line. You may want to replace the horizontal lines with lines drawn in light grey rather than remove them altogether in order to improve accuracy of reading the Y-axis (if this is important).

Discussion on Activity 6

In the first dialog box, light grey lines are used to enforce grouping. Proximity is therefore used to group functions, which have a similar task, or work on the same object (file, clipboard etc.) Shape similarity is also used for the table tool. Word allows the insertion of a Word table or an Excel table. The icons are identical, except the Excel table has the Excel logo placed on it.

In the second dialog box, all the functions act on the image. Shape is used to show similarity between the picture function buttons and functions (e.g. the OUT and IN options). In this case the colour of the buttons is purely aesthetic.

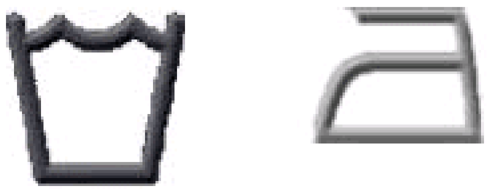
Discussion on Activity 7

The usual affordance is underlined text in blue. Sometime web pages contain underlined text, which is not a link – this is a poor affordance. Also, web page authors change text colour so that you cannot be sure blue text is a link – again, a bad affordance.

Pictures which link are also confusing as picture which are not links look identical to pictures which are links. One way to improve affordance is to use “roll-overs.” Rollovers are images which change when the mouse rolls-over them.

Of course, if all else fails, look at the mouse pointer. When it is an arrow, there is no link. When it is a little hand, the object under the pointer is a hyperlink.

Discussion on Activity 8



Discussion on Activity 9

The menu is here as people in western culture start reading from the top left hand corner and proceed across the top until moving on to the next line. Someone from China on the other hand, reads in columns, so it might be more sensible to put the menu down the left edge of the screen.

Chapter 5. Usability Implications of Perception and Memory - Introduction

Table of Contents

Context

Unit 4 introduced the area of cognitive psychology and focused on the important cognitive process of visual perception. Although vision is, at present, the key sense used in interaction, there are and will be roles for our other #input channels#. This unit looks at the user#s capabilities and limitations in terms of the auditory (hearing) and haptic (touch) senses and also, more briefly, at the possibilities of using taste and smell in interaction. Technologies and applications relevant to the different channels will be discussed. The unit continues with an exploration of two other cognitive elements: the process of attention and human memory. You will learn about these elements and, importantly, how this knowledge can be applied in the design process.

Objectives

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

- Discuss the use of audio at the interface.
- Illustrate how current audio use might be extended both verbally (speech synthesis and recognition) and non-verbally (e.g. auditory icons and earcons).
- Discuss the current limited role of the haptic sense and possible extensions (e.g., Braille output, use in virtual reality etc).
- Give example potential applications and technologies for smell and taste output / input (e.g. e-commerce).
- Define the cognitive processes of attention.
- State why designers should accommodate a user’s attention mechanism.
- Show how designs can be improved in terms of the way they focus a user’s attention.
- Distinguish between the 3 forms of human memory.
- Show how knowledge of the 3 forms of memory can be used in design.
- Distinguish between the process of recall and recognition and discuss why and how recognition can be promoted.

Introduction

The previous unit discussed the properties of the visual medium and the implications of what we know about human vision for the design of interactive systems. Of course, vision is not the only way we are able to perceive the world around us, and in this section we shall discuss the properties of some of our other senses, especially hearing, and the implications that these properties have for user interface design.

In addition to considering how senses other than vision may be used in human-computer interaction, we will also look at the mechanisms by which sensory inputs are processed by the human cognitive system. In particular, we shall look at how we are able to pay attention to certain sensory stimuli and mental activities, while ignoring others, and what implications this has for interface design. Finally in this unit, we will outline how human memory works, and once again take a look at the relevance this has for interface designers.

Sound in the Interface

The vast majority of computer-based user interfaces that we encounter rely almost totally on the visual medium. However, another medium that is frequently used in interface design is sound. Before discussing how interface designers do and may make use of sound, we will review some of the properties of the audio medium consisting of sound together with the human sense of hearing.

Properties of sound and hearing

A number of properties of the audio medium are pertinent to the use of sound in the interface.

Sounds can vary in a number of dimensions: pitch (or frequency), timbre (or musical tone), and intensity (or loudness). Not all sounds or variations is sound are audible to humans. The ear is capable of hearing frequencies ranging from about 20Hz up to about 15KHz, and differences of around 1.5 Hz can be discerned (though this is less accurate at high frequencies).

The capability of computer sound output devices to produce variations along each of the dimensions of pitch, timbre and intensity means that sound output is potentially a rich and sophisticated medium for conveying information to users.

The audio medium, like any other, has a number of inherent properties that constrain the way humans process and make sense of the sounds they hear. Therefore, understanding these constraints will be crucial to the successful use of sound in interactive systems.

In contrast to vision, sound is a "volatile" medium in the sense that sounds do not persist in the same way that visual images do. Or to put it another way, the visual field can be regarded as a parallel array of information elements, each of which may remain constant or may vary over time. Sound, on the other hand, can be seen as a single element (described by its pitch, timbre and intensity) that may vary over time, and the rate at which it varies or carries information, or is perceived, is not under the control of the listener. Its potential as a means of conveying information, and the amount and type of information that can be carried, is therefore rather different from that of the visual channel. Consequently, the visual channel can be regarded as frequently having a much faster access time. For example, a large amount of information may be made simultaneously available in the visual channel, whereas presenting the information in the audio channel may take longer as the information must be "serialised". A further implication of this is that an audio information stream may place greater demands on the user’s memory: while listening to part of a message, the user must remember what has gone before. When reading a visual display, the parts previously read remain visible.

Another relevant property of the audio medium is the fact that, unlike vision, hearing is non-directional. While binaural hearing does grant us a limited ability to determine the direction of a sound source, there is no sense in which we can listen in a particular direction. Similarly, while we can very easily control our visual sense (e.g., by looking in a particular direction), it is much harder to be selective about what one listens to.

It is well known that people are rather good at noticing changes in sound – such as the onset or cessation of a noise or tone. However, we are rather less good at extracting information from a stream that remains relatively constant. In fact, if a background sound remains relatively constant, over a period of time we will tend to become less aware of it, and eventually will filter it completely and cease to notice it at all. See the later section on ATTENTION.

A further property of sound and hearing that designers should be aware of is that we are relatively poor at separating out a single sound or sequence of sounds from a background of similar sounds. Imagine trying to hold a conversation in a noisy environment where many other people are talking. Or trying to follow two conversations at once.

For most of us, in our everyday lives, sound plays a very important part. It is often said that the majority of information we receive about the world comes to us through our visual sense. While this is true, it is also the case that sound plays a central role in communicating with others (through speech and other sounds), receiving information and entertainment (through radio broadcasts, musical performances, and so on), and allowing us to be aware of events – some of which may be outside our visual field (police sirens, ringing telephones, etc). Sounds that are apparently in the ‘background’ often give us vital clues about the status of ongoing processes (e.g., the sound made by a car engine as we are driving, the noise made by machinery in a factory).

Despite the apparent limitations described above, sound is a remarkably important channel for conveying information. Next we will look at some of the ways sound is used in current user interfaces, and how it could be used in the future, and will identify some guidelines that can help designers to make the use of sound more successful.

Activity 1 - Sound and vision

You can carry out this activity on your own if you like, but it might be easier if you can work in pairs. Compare the time taken to read a written passage, with that taken to listen to the same text being spoken. For the latter part, you can simply read the text out loud, or use a computer speech synthesiser.

Activity 2 - Menu Structure

It is often said structured menus should be made wide (many choices in a menu) and shallow (few levels of menu), rather than deep (many levels). Studies suggest that doing this will improve allow users to carry out tasks faster, while making fewer errors. See (Shneiderman, 1998, pages 249-250) or (Norman, 1990, chapter 5) for discussions.

Suppose that instead of menus on a visual display, we are designing menus in an audio interface for telephone-based services. Do the same guidelines about wide/shallow or deep/narrow structures apply? How would you investigate the efficiency of different audio menu schemes?

A Discussion on this activity can be found at the end of the chapter.

Use of sound in current and future interfaces

Sound is currently used in many computer interfaces and other interactive devices, sometimes as an important source of information, and at other times simply as a means of making the interface seem more impressive. Sound can be used for both input and output functions in an interface, and the kinds of sounds that are used include abstract tones and bleeps, naturally occurring sounds, music, and, of course, speech.

The audio channel is, for most of us, a rich and important source of information in interactions with other people or with the environment. However, sound has played a more limited role in human-computer interaction than in other aspects of life. Part of the reason for this may be that it is not always clear what kinds of functions sound is appropriate for, and how to make effective use of sounds to support those functions.

Audio Alerts

While sound might be ineffectively exploited, that is not to say it is unused. Almost all computer systems "bleep" when an error of some sort occurs. What is the purpose of such a bleep, and what can the user infer from it? Clearly the sound indicates to the user that something has happened, but it is typically left up to the user to determine by other means the nature of the event (an erroneous input, or a system generated warning, or simply the arrival of email?), its source (one of several applications, the operating system, networking software?), and what should be done about it. The following, rather extreme, example highlights the kind of problematic situation that can arise.

Note

In 1979, at the Three Mile Island nuclear power station in the USA, one of the most serious nuclear accidents in history took place. As the incident was unfolding, operators in the plant control room were faced with a bewildering array of instruments and displays giving information about the state of the power station. As problems arose, auditory alarms sounded to alert the operators to what was going on. The following commentary on the incident (drawing on the report of the subsequent investigation by a Presidential Commission) explains one aspect of the operators’ predicament:

Frederick and Faust [two of the operators] were in the control room when the first alarm sounded, followed by a cascade of alarms that numbered 100 within minutes. … Later Faust would recall for the Commission his reaction to the incessant alarms: "I would have liked to thrown away the alarm panel. It wasn't giving us any useful information."

Partly as a result of the confusing alarms, the operators failed to diagnose one of the serious problems with the reactor for two hours, during which time extensive damage and leakage of radioactive material occurred.

The point is that although alarms and alerts can be successful for indicating that some event has occurred, they often carry little useful information about the nature or location of the event or what should in response. And if several notable events occur together, then providing an auditory indication of each is simply going to confuse users. The world of power stations and control rooms may seem very far removed from everyday design, but the same issues are relevant in the interfaces of desktop systems and the design of web pages.

Sound output, therefore must be used with care. Sometimes it is appropriate to indicate a change of status or a particular event with a sound, but we must be aware that beeps and similar alerting sounds often provide the user with too little information about either the nature of the event or what action will need to be taken as a result. Used with care, however, sound can enhance an interface and provide users with an important source of information.

Providing Information

So far, we have been discussing sounds without saying much about what kinds of sounds are typically – or in the future could be – useful features of an interface. The simplest and most basic sounds are simple bleeps that indicate an event or change of state. However, other possibilities exist.

An idea that several researchers have experimented with is to add sounds to many of the familiar features of existing visual user interfaces. So, in addition to the visual cue provided by an icon, an auditory cue would be provided as a further memory aid. The sounds used in some of these experiments were natural ones, chosen to match the kind of interface objects with which they are associated. For example, folders on a desktop might have the sound of paper crumpling (presumably because in the real world, we put paper into folders), and dropping an item in the wastebasket might produce a sound of breaking (because things sometimes break when we throw them in the bin).

While this might seem an attractive way of providing the user with additional feedback and extra ways of remembering how the interface works, it has some problems. One is that the same sound might mean different things to different people – so the sounds are not as "natural" as they first seem. Another problem is that while auditory equivalents may readily be found for some interface elements, there are a great many computer-based objects and operations for which no real-world audio counterpart exists.

One researcher in this area, Stephen Brewster, has taken up the idea of augmenting existing, visually based interfaces with sounds to assist the user, but has used abstract sounds rather than naturally occurring ones. These sounds – known as earcons (icons for the ear!) – are made up of short musical phrases that vary in the sequence of notes, overall pitch, tempo, and so on. Earcons have been added to icons, menu items, and so on, of conventional computer interfaces.

Speech Output

In communicating with other people, we most commonly make use of speech, and our interactions with computers can be similarly speech-based. Speech synthesis has been possible for quite some time, but it is only relatively recently that using synthetic speech has become a reality in everyday interfaces. However, speech is becoming increasingly popular ad an addition to more conventional user interfaces.

Note

Example: Voice as an alert

Recent versions of Apple's MacOS operating system include a speech synthesiser that will speak the contents of pop-up alert windows. If a pop-up window appears with a text message, this will be accompanied by a voice speaking the text.

While this will undoubtedly be a valuable feature for some users, others simply find it unnecessary, and turn it off (as we already discussed, it can take rather longer to listen to the verbalisation than it does to read the corresponding text). In fact, in Apple's implementation of this feature, there is a short delay between the pop-up box appearing and the start of the speech, and if the pop-up is closed or cancelled during this delay, then the speech is never started. Some users are quite surprised when, the first time they are slow to close the dialogue box, the computer starts speaking to them!

Note

Example: Voice can reduce demands on the visual sense

On modern aircraft flight decks, synthesised speech is used for many applications. For example, in the final stages of an approach to landing, the aircraft's height above the ground is critical. On some aircraft, the computer system monitors the height and provides a verbal read-back of the height in intervals of 10 feet as the aircraft approaches the ground.

This is one instance where speech has the potential to add real value to an interface. On approach to landing, a pilot's visual sense is usually in high demand – looking out of the window at the runway below, as well as monitoring other instruments. If the critical height information were available only on a visual display, then the pilot's vision would be stretched even further, by the requirement to look at yet another display, and by the need to continually re-focus on different things. On the other hand, if some of the information can be "off loaded" and presented acoustically instead of visually, then the demand on the pilot is lessened, reducing the potential for error and catastrophe.

Sound Input

Sound can also be used as an input device. Almost always in human-computer interaction, this means using voice recognition – a relatively new technology that is beginning to achieve the level of performance needed to be effective. One often hears a view, presumably inspired by science fiction movies, that if only we could just speak to our computers, then all our usability problems would be solved. Indeed, we would no longer need user interfaces! It must be emphasised, though, that voice recognition is yet another kind of user interface technology, and just like the mouse and keyboard. It therefore is appropriate for some things and not for others, and using voice input has just as much (or perhaps more) potential to create usability problems as any other technology.

A number of products exist that allow standard desktop computer systems to take speech as input. For instance, recent versions of Apple's MacOS operating systems allow the user to speak standard commands (such as Save, Print, Open, and so on), which the computer interprets and executes. Typically, though, users spend only a small part of their time entering commands into a computer, and much more time entering and editing the content of their documents, so a facility for speaking commands is only of limited use.

A number of products are available that allow users to speak text to the computer, which is then used as the input for word processors and other software.

While voice input might be a nice addition to existing desktop computer systems, it is likely to play a much more central role in interactions where using familiar visual output and mouse and keyboard input is difficult or infeasible. One such situation that we have mentioned before is the provision of services that will be accessed by telephone using only the auditory channel. Several products exist that allow designers interactive telephone-based services to incorporate voice recognition technology. This allows users to provide information, commands and so on. For example, the Vocalis group markets a range of products and has made demonstrations available over the telephone.

Implications for user interface design

In the following sections we will present some suggestions and guidelines for when and where sound might most successfully be employed in the user interface for both the output and input channels.

Sound output

Audio output may be the appropriate channel to use for:

- Giving immediate feedback that an action has taken place (buttons on ATMs, telephones, etc., that "beep" or "click" as they are pressed);

- Presenting different kinds of information to that made available using the visual channel (e.g., non-static information such as alerts);
- Augmenting visual interfaces by providing additional information and cues
- Supporting users for whom the visual interfaces are not an option (e.g., those with visual impairments)
- Supporting users whose visual senses are already heavily used for other parts of their task (e.g., aircraft pilots)
- In interfaces where visual information cannot be presented (e.g. mobile or handheld devices with small or no screens).

Audio output may not be particularly good for:

- Constantly changing status information
- Use in shared offices or workplaces, where privacy is important or where the output of many users' computers would lead to confusion or disturbance
- Noisy environments where sound may be difficult to hear (e.g., the user interface of machinery on a building site)
- Quiet environments where sounds could cause distraction (e.g., a library or recording studio)

Voice input

Sound can also be used as an input device. Almost always, this means using voice recognition – a relatively new technology that is beginning to achieve the level of performance needed to be effective.

Voice input may be appropriate for:

- Users who are unable to use more familiar input devices like the mouse or keyboard (e.g. those with severe motor impairments)
- Users who are busy doing other things (e.g., an aircraft pilot, whose visual channel is occupied monitoring the approaching runway during landing)
- Interactions that may be cumbersome using the other interaction devices that are available (e.g., menus used in mobile phone services).
- Small and pre-determined range of spoken commands.

Voice input may not be particularly good for:

- Noisy environments
- Use by many different users with different voices and accents
- Wide range of words or a specialised technical vocabulary. Note that some commercially available voice recognition systems have versions adapted to technical specialisms such as medicine or law.
- Things that are not easily verbalised (e.g., diagrams)

Other Senses, Other Devices

So far, we have looked at how sound and vision can form parts of the interface between user and computer. We now ask the question of whether some of our other senses and physical abilities can be used in order to facilitate human-machine communication. The use of other senses is usually proposed for one of two reasons. The first is to replace a sensory channel that is more often employed in user interfaces. This is typically so as to make a system accessible to users with sensory impairments. For instance, if touch can be used to do some or all of the work of vision, then a system will be more accessible to blind users. The second reason is to add to or augment the experience created by senses already used. For example, if the web site about cookery were able to allow users to smell the results of cooking, the effect might be a richer and more enjoyable experience for the user.

Touch

In everyday life, we use touch, and related senses (that give us information about temperature, texture, pressure, body position and orientation, and so on) as yet another way of gaining information and feedback about objects and events around us – especially those with which we are interacting most directly.

Currently, little use is made by interface designers of users' sense of touch. However, a number of experimental prototypes and products meeting specialised requirements serve as excellent examples of what is possible and what may become more significant for interface design in the future.

Several devices have been designed that allow the computer to produce output in a form that can be sensed by touch and related senses. Such devices seem to be most successful is in providing output that is tactile (i.e., that is amenable to the sense of touch – especially by having distinctive texture or shape) and providing force feedback (where the user experiences the computer output as a force on their body). In some cases, these devices are intimately connected to the input devices through which the user issues commands to the computer, and are therefore capable of providing instant and very direct feedback to the user.

Tactile Output

Output devices exist that convert the output of the computer into tactile form. For example, the text normally displayed on a computer screen can be rendered in a tactile language, such as Braille, making it available to users with vision impairments.

For example there is product called "Braille ‘n' Speak" that allows blind or partially sighted users to take notes using a specialised personal organiser. Notes can be "read back" via either a synthesised voice or a "refreshable Braille cell" that turns text into tactile form.

Tactile output devices of this type essentially use touch as an alternative output medium. Other devices rely on an observation about the way most human interaction with the physical world takes place. The separation between input and output that is so clear-cut in many conventional computer interfaces is not so sharp in other contexts. As we take an action in the real world (such as taking hold of a cup and lifting it up) we get immediate feedback (about the texture, temperature, and weight of the cup), through the sensation of pressure in our muscles, joints, and fingers. Thus the actions we take and the feedback we get as a result are very closely connected.

A simple but effective way of combining input and output for user interfaces has been to allow a pointing device such as a mouse or joystick to provide physical feedback my making it resist the user's actions in a context sensitive way, and provide tactile feedback as the mouse pointer is moved over screen objects.

For example, the Moose is a mouse-like device that gives various kinds of physical feedback. If the user is dragging an object, the mouse will feel heavier than normal, and the user will be given tactile cues as the pointer moves over different kinds of screen object (for instance, moving across the edge of a window could produce a "click" sensation).

Such force-feedback mice are a relatively simple way of combining user input and feedback, but more sophisticated products exist. For example, the Phantom allows the user to move a pen-like stylus through space in order to interact with a computer using gestures. Feedback is provided by mechanisms attached to the stylus that make it easier or harder to move, or simulate the effects of different objects and textures that it comes into contact with.

A similar input/output technique that has been developed for use with virtual reality systems involves the use of a "data glove", which senses the position, orientation, and movement of the user's hand, allowing gestures to be used as inputs, and objects in a virtual reality system to be grasped. Several products exist that add force, pressure or vibration feedback to a data glove, allowing the user to "feel" virtual objects they touch or grasp.

Other Senses

So far, we have discussed how the senses of vision, hearing and touch might be employed by user interface designers to provide a richer and more compelling experience for the user. But that still leaves several senses that have not been utilised in human-computer interaction. The senses of smell and taste are unlikely to be very effective as ways to convey large amounts of structured information, but both these senses are highly evocative and profoundly shape our experience of real-life situations.

Surely, though, smell is not a feasible medium for computers to produce their output? At least one company is developing a product that allows computers to generate olfactory output: you can read about the "iSmell" device in an article in Wired magazine [http://www.wired.com/wired/archive/7.11/digiscent_pr.html].

Activity 3 – Smell, taste and touch in the interface

It is clear how designers can make use of user's auditory and visual senses in their designs. Assuming the necessary hardware was available, how could the senses of touch, smell and taste be used as well as vision and audition for e-commerce applications over the Internet?

A Discussion on this activity can be found at the end of the chapter.

Multiple modalities

We have now discussed a number of modalities – or human sensory channels – that interface designers routinely require users to make use of (and a couple of modalities that aren't yet used in human-computer interaction, but may one day be). Each of these has properties that make its use more or less suitable for particular kinds of function in particular kinds of situation. A design challenge that we haven't mentioned yet, however, arises when more than one sense or means of providing input to the computer is employed in an interface in a way that requires the user to use them in co-ordination.

Several systems have been developed that allow the user to mix commands expressed in various ways. One example is MATIS, a system for finding information about air travel. The user may specify their constraints (such as origin and destination, date of travel) using a mixture of typed information, pointing with the mouse, and speech. The design challenge for such a system is to be able to correctly interpret a command expressed in a simultaneous mixture of these forms. So the user may specify travel by speaking "from here to there", while pointing at origin and destination cities with the mouse, and the system must match the occurrence of words like "here" with the corresponding mouse actions.

Not all users are the same!

As a final note in our discussion of perception, it should be emphasised that not all of our users have the same perceptual capabilities.

Graphical user interfaces are – not surprisingly, given the name – highly visual in the demands they make on users. Much of the user interface technology that is being designed for personal computers and used on the world wide web makes strong assumptions about the perceptual capabilities of users, and designers are therefore often excluding sections of the population whose vision, hearing, or other senses and abilities are impaired. Aside from any moral considerations about making technology accessible to all, there are good business reasons for considering access to our designs: the more people who are unable to use out designs, the more potential customers we have lost. This is especially true on the web, where a large number of potential customers will look at a site, and decide whether to continue to use it on the basis of whether its design helps them to carry out their tasks easily.

Sometimes, designing for inclusiveness and accessibility isn't only an ethical consideration or a matter of commercial good sense. In some cases it is also a legal requirement. For example, in the United States, the Americans with Disabilities Act (ADA) requires that certain organisations and bodies (particularly governmental bodies) provide "effective communication" when they use the Internet to make information and services available. Specifically, this ruling notes that web sites must be accessible to those using screen readers and receiving web information through auditory rather than (or in addition to) their visual sense.

Review Question 1

What are the properties of the audio channel that make it different from vision? What are the implications for the use of sound in the user interface of ordinary desktop computers?

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

Review Question 2

Most telephones in the UK provide a facility for users to find the number of the last person who called them. By dialling 1471 a computer synthesised voice informs the user of the phone number and the time at which the call took place, for example, "Telephone number 07934 363 001 called today at 0930 hours.." Mobile phones, on the other hand, can show the numbers of the last several callers on the display. Compare the two presentations of roughly the same information and identify some of the advantages and disadvantages of using the auditory and visual senses for presenting this kind of information.

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

Review Question 3

Voice synthesisers provide an obvious way for blind and partially sighted users to browse the web. What might some of the problems be with this way of making the web more accessible?

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

Attention

The previous section and the previous unit have discussed some of the ways that humans are able to experience the world they inhabit using their visual, auditory, and other senses, and the importance of understanding the properties of these senses when designing user interfaces. It is clear that the world presents an extremely rich and complex set of stimuli to the senses. It is only by filtering out some of this, and ignoring the rest that we are able to make sense of the world around us, instead of being hopelessly confused by it. The processes by which humans are able to focus on some things and filter out others, to concentrate on certain mental or perceptual events and devote less mental resource to the rest, are known collectively as attention.

Attention is..

Every one knows what attention is. It is the taking possession by the mind in clear and vivid form, of one out of what seem several simultaneously possible objects or trains of thought. Focalization, concentration, of consciousness are of its essence. It implies a withdrawal from some things in order to deal with others.

This oft-quoted passage from one of the founders of modern psychology captures some of the essence of what we mean by attention. At the heart of the concept is the idea that there are limits to our cognitive and perceptual capabilities, and that multiple demands are almost always made on these limited resources. Some form of "filter" or selection process is required so that we can focus on only a subset of the possible mental and perceptual activities; this process of focusing is known as attention.

We are able to choose a single stimulus from the many of offer on which to focus on using what is often referred to as selective attention. However, we can also engage in a number of activities and focus on a number of things simultaneously, in which case out attention is said to be divided. The selection of what to attend to may be the result of conscious deliberation, or deciding to concentrate on something specific, or may arise from unconscious or involuntary responses to things that "grab our attention".

Much of the psychological research on attention has focused on the ways that people are able to direct their attention to particular perceptual events in their auditory experience. However, many of the points that are made and theories that have been developed can be generalised to other senses and other mental processes.

The consequences of inattention

We can fail to pay attention to the appropriate thing for a number of reasons, and the consequence is likely to be that we fail to complete a task as we intended, or substitute an incorrect task instead.

Does the following story, as told by psychologist James Reason, sound familiar?

Note

Imagine you have a visitor who has requested a cup of tea, while you only drink coffee. You go to the kitchen intending to prepare both tea and coffee, but return with two cups of coffee. The reason for this slip is clear; you failed to make an attentional check on your plan at the point where the initial common pathway, boiling a kettle, branches into its separate tea- and coffee-making components. As a result, you proceed along the habitual coffee route.

In general, a number of possible consequences can result from inattention. The kind of unintentional error or "slip" exemplified above, where through a failure to pay attention, a person carries out a more familiar task instead of a correct, but less familiar one, is known as a capture error. Related to this is a memory lapse, where something is simply forgotten because we aren't paying attention. Common examples are using a photocopier, but failing to remove our original document at the end, or using a vending machine, but failing to take the change.

A third class of attention-related problem occurs when we are simply distracted from one task by some other stimulus or task. An example when we stop working at the computer to answer the telephone. Sometimes the distraction is unwanted and unnecessary (such as the distraction caused by blinking text on a web page) and sometimes, even if the distraction is desirable (such as an important telephone call) it creates problems when we attempt to resume the original task.

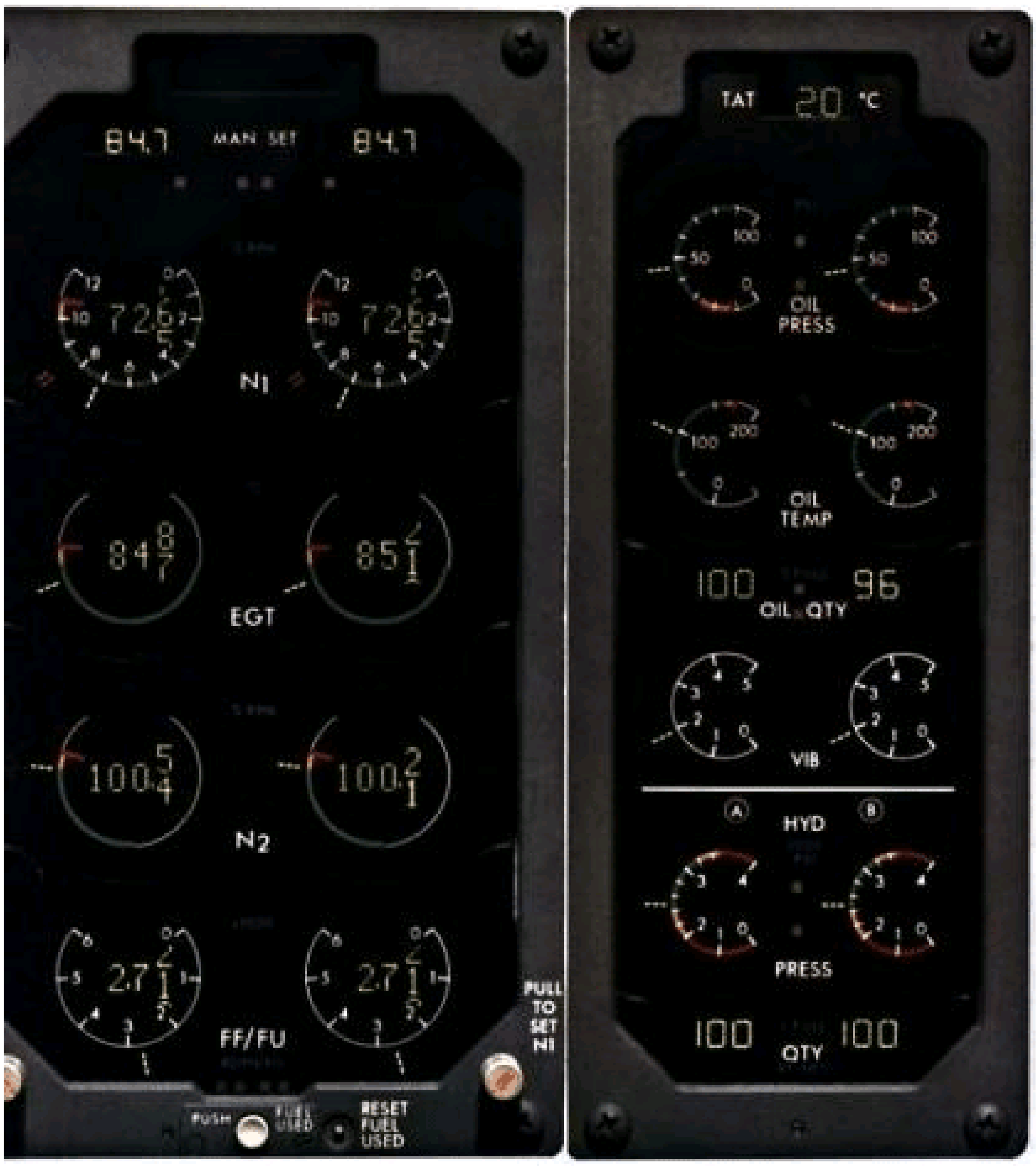
A further kind of problem arises if our attention is not focused on something important. For example, we can miss the arrival of new e-mail unless we happen to be attending to the icon that announces its arrival. Of course, the effect of failing to attend to the right thing is not always so trivial, as the following example illustrates.

Note

A serious air accident resulted when the pilots of an aircraft incorrectly located the source of an engine problem their aircraft was having as being in the right engine (it was actually in the left engine). The pilots shut down the healthy engine and the aircraft subsequently crashed as it was attempting to land.

An instrument on board the flight deck could possibly have revealed the source of the problem, had the pilots noticed it. It is, of course, all to easy to blame the pilots for making an incorrect decision, but it is much more worthwhile to look at how attention getting characteristics designed into the user interface had a part to play.

The relevant instrument (lower right, underlined in white, in the picture below) was part of a panel containing several other dials, itself part of a mass of controls, dials, instruments, and computer screens on the flight deck. The dial in question has no alert or other mechanism to attract the pilots' attention to the fact that it was displaying an out-of-normal-range value. In fact the normal and abnormal ranges are not even shown on the dial at all (e.g., with a "red zone", as on many of the instruments) Therefore, there was nothing to encourage the pilots to focus on this item in preference to all the other possibilities.



Note that the contribution of the dial design was only one component in a complicated series of occurrences that lead eventually to the accident, which is described from a different perspective in Unit 2.

Factors that affect attentional focus

As has already been said, at the core of the concept of attention is the idea that we have only a limited capacity for perceiving the world and carrying out mental and physical tasks. We therefore need choose from the large number of competing possibilities, a small number of things to concentrate on. A number of factors connected with the stimuli we receive, the way our mental apparatus processes them and our state of mind at the time affect the way we are able to achieve this focusing.

These include the following properties of the stimulus

- Movement and change – the human eye is able to discern great detail, but only in the central part of the visual field. In other parts of the visual field (the "peripheral vision"), we are not particularly good at seeing detail and colour, but are very good at spotting movement and change towards which we can direct our attention. This is why it is possible to notice that something is happening "out of the corner of ones eye", without seeing exactly what.

- Colour, size, intensity– certain colours tend to be particularly good at attracting our attention. Similarly if one object is physically larger, brighter, bolder or more visually intense than surrounding ones, then it is more likely to attract attention
- Number of competing stimuli – the greater the number of possible things to attend to, the less attentional resource there is to devote to each.

In addition to these properties of stimuli, two factors concerning the human observer have an impact on how and where attention is directed.

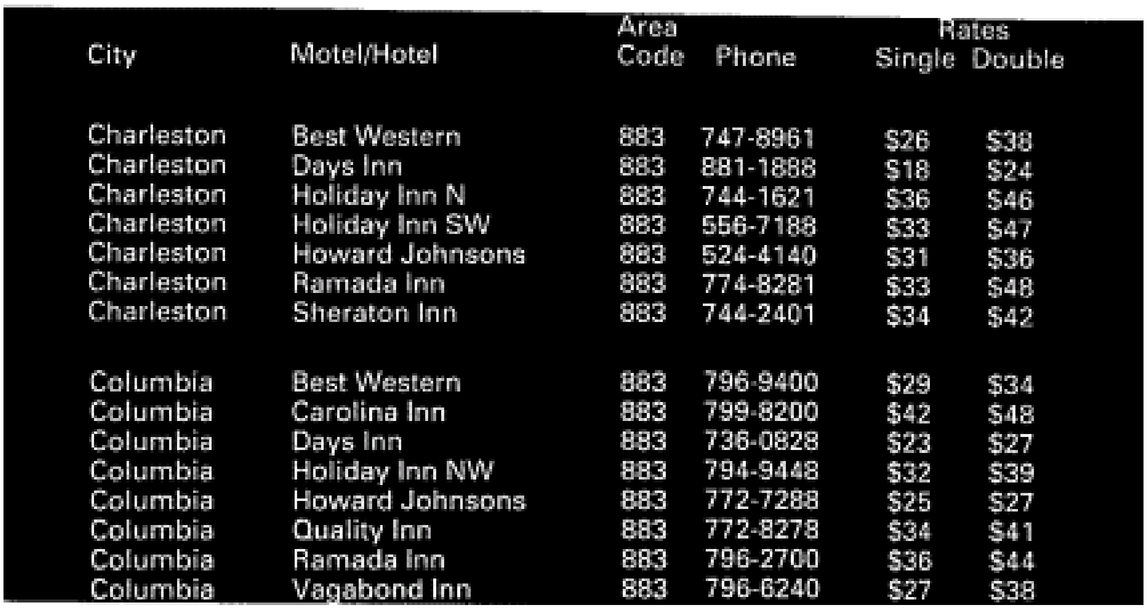
- Meaningfulness – we are more likely to focus on things that are meaningful or understandable or make sense to us. One aspect of this is a phenomenon sometimes referred to as the "cocktail party effect" Imagine you are at a party or in some other situation where a crowd of people are talking. There is a constant level of background sound, yet it is possible to pick out words and threads of other conversations across the room. We are especially good at doing this if the content of the conversation is meaningful to us – for instance, if we hear our name mentioned in someone else's conversation, we can "tune in" and listen to what is being said.
- Emotional state – in a particularly stressful or pressured situation, a person's attentional capacity tends to be diminished, meaning that they are able to focus on fewer things at once. This fact may not be very significant for the design of web sites and desktop systems. However it is highly relevant for the design of systems such as aircraft flight decks. If designers are not careful, there is a tendency to place the greatest attentional burden (e.g., lots of visual tasks to be carried out) on the user at precisely the most stressful times (e.g., landing the aircraft).

Activity 4 – Structuring information and focusing attention

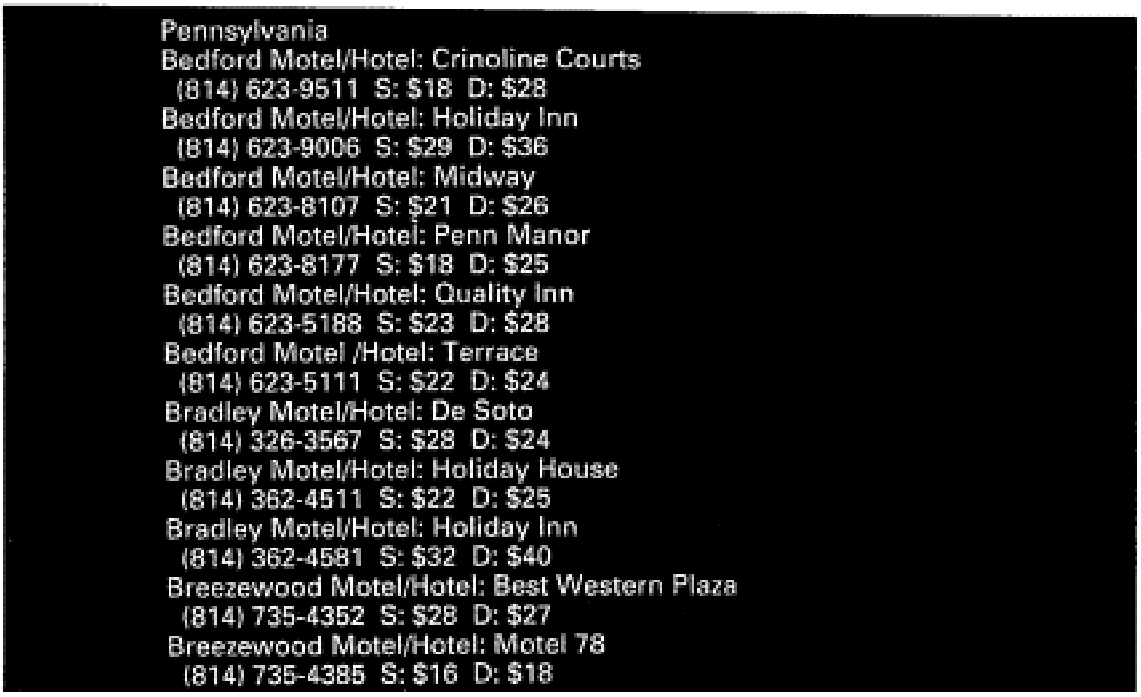
The following exercise is taken from Preece et al. (1994), pages 102-103.

You can carry out this exercise by yourself, but it will be easier and more reliable if you are able to get someone else to help by carrying out the timing. The two figures below are alternative presentations of the same kind of data. With the first display find out how long it takes you to find (1) the phone number of Howard Johnsons in Columbia and (2) the name of a motel with a double room for $46. With the second display find out how long it takes to find (1) the telephone number of a Holiday House and (2) the name of a hotel with a double room for $27.

Which display takes the longest? Why do you think this is?



City	Motel/Hotel	Area Code	Phone	Rates Single	Rates Double
Charleston	Best Western	883	747-8961	$26	$38
Charleston	Days Inn	883	881-1888	$18	$24
Charleston	Holiday Inn N	883	744-1621	$36	$46
Charleston	Holiday Inn SW	883	556-7188	$33	$47
Charleston	Howard Johnsons	883	524-4140	$31	$36
Charleston	Ramada Inn	883	774-8281	$33	$48
Charleston	Sheraton Inn	883	744-2401	$34	$42
Columbia	Best Western	883	796-9400	$29	$34
Columbia	Carolina Inn	883	799-8200	$42	$48
Columbia	Days Inn	883	736-0828	$23	$27
Columbia	Holiday Inn NW	883	794-9448	$32	$39
Columbia	Howard Johnsons	883	772-7288	$25	$27
Columbia	Quality Inn	883	772-8278	$34	$41
Columbia	Ramada Inn	883	796-2700	$36	$44
Columbia	Vagabond Inn	883	796-6240	$27	$38



Pennsylvania
Bedford Motel/Hotel: Crinoline Courts
(814) 623-9511 S: $18 D: $28
Bedford Motel/Hotel: Holiday Inn
(814) 623-9006 S: $29 D: $36
Bedford Motel/Hotel: Midway
(814) 623-8107 S: $21 D: $26
Bedford Motel/Hotel: Penn Manor
(814) 623-8177 S: $18 D: $25
Bedford Motel/Hotel: Quality Inn
(814) 623-5188 S: $23 D: $28
Bedford Motel /Hotel: Terrace
(814) 623-5111 S: $22 D: $24
Bradley Motel/Hotel: De Soto
(814) 326-3567 S: $28 D: $24
Bradley Motel/Hotel: Holiday House
(814) 362-4511 S: $22 D: $25
Bradley Motel/Hotel: Holiday Inn
(814) 362-4581 S: $32 D: $40
Breezewood Motel/Hotel: Best Western Plaza
(814) 735-4352 S: $28 D: $27
Breezewood Motel/Hotel: Motel 78
(814) 735-4385 S: $16 D: $18

A Discussion on this activity can be found at the end of the chapter.

Implications for user interface design

In order for a user to be able to successfully use a computer system, it is imperative that they be able to direct their attention to the right thing at the right time. In the remainder of this section, we outline some of the measures that designers can take to help users to focus their attention on the parts of the user interface relevant to their task, and to remain focused and not be unnecessarily distracted.

Design features affecting attention

Let us take as an example the design of web pages. The kind of question a designer or usability analyst should be asking is: what does the user need to focus on in order to carry out their task? is the user's attention likely to be focused on these things? Or are there features of the design that tend to encourage them to focus elsewhere? If the user does get distracted and focus their attention elsewhere (sometimes this is unavoidable, and results from events outside the user interface – such as the telephone ringing or a colleague walking into the office), can they pick up the task again later and continue by re-focusing their attention appropriately?

As designers, there are a number of techniques we can use to try and ensure that users' attention will be directed towards the appropriate things for the task at hand. Although these are exemplified by pages and sites on the World Wide Web, the points made apply equally to other interactive systems.

- Flashing, motion, change – these techniques may be a powerful way to attract attention, but they can often simply be a distracting annoyance to users – see the section below on attention grabbing versus helping the user to decide.
- Difference – differences in colour, size, shape, intensity, and so on, can be used to make some items of a display more attention getting. For example, colour can be used to segment the page into regions that help users to focus, and to highlight important items.
- Meaningfulness and importance. Users will tend to find it easier to focus on items that are easy to understand and important to their task.
- Visual structure and grouping of items meaningful units can help the user to focus on the items that are relevant to their concerns. See for example BBC News website [http://news.bbc.co.uk/], where a large amount of information is presented in a structured way.

- Simplify the choices presented to the user. Look, for example, at UCT's Department of Computer Science [http://www.cs.uct.ac.za]
- Alerting mechanisms such as animations, and pop-up screens should be used with care, and only to provide warnings or supplementary information.
- Spatial cues, indicating, for instance, where in a fill-in form the user currently is, can help the user focus on the parts of the a task they are currently working on.
- Temporal cues, indicating where in a process or sequence of sub-tasks the user currently is. See, for example, the sequence of steps involved in purchasing an aeroplane ticket at Mango Airline [http://www.flymango.co.za]. This way of depicting the steps in a process is becoming increasingly common in e-commerce sites.

Activity 5 - Attention focusing in Web design

Look at the following collection of web sites, and for each, consider how the page designer has built in features that can help or hinder the user in their ability to focus their attention on what's important, and to remain focused on the task at hand.

- http://www.bbc.co.uk/
- http://www.amazon.com
- http://www.telkom.co.za

Attention grabbing versus helping the user to decide

It is all very well to design user interfaces with features that attract the user's attention in one way or another, but does this really contribute to the usability of the interface?

A typical situation, which might be familiar to users of the web, is that the designer includes features that are likely to attract the users attention, presumably because the designer felt that certain parts of the interface, or certain pieces of information were especially important. If you are a designer, this might seem quite easy to do: simply decide what you what the user to look at, and make it larger than surrounding elements of the interface. Or make it more brightly coloured, or make it move or flash. It you really feel that the item is very important (e.g., if its an advert for a product the user hadn't yet realised they wanted to buy), then use all of these techniques together, and include a large, attractively coloured animated graphic in the interface. Surely, the user will be virtually unable take their eyes off the graphic and will soon forget what it was that they originally came to your site for!

Actually, if this technique does work at all, and the user's attention is captured, they are likely to find it distracting and annoying. Users are likely to stop using the site and choose a different one that allows them to carry out their task without being bothered animations and so forth.

A better alternative might be to accept that the user might be best placed to decide what is important and what is not for purposes of the task that they are trying to accomplish. The design should therefore try to support users in identifying what possibilities the interface is offering them, assessing their importance and relevance, selecting an appropriate alternative, and remaining focused on their task without unnecessary distractions.

In any case, recent research suggests that crude attempts to attract web users' attention are less than successful. A study in which eye-tracking technology was used to find out where web site users direct their gaze showed that users tend to prefer looking at text, and avoid looking at images and animations. Even when the graphics contained useful information! A finding that suggests that users have become habituated to web designers tricks and have trained themselves to overcome basic physiological and psychological responses, so as to ignore web features they imagine to be superfluous. This tends to indicate that banner advertising and other attempts to encourage the user attend to things that may not be relevant to their interests and intentions may be far less effective than designers would like to believe. It should be emphasised that this research finding is specific to the users of particular kinds of web site, and cannot necessarily be generalised to other settings and other kinds on user interface.

Of course, if the user interface emphasises nothing in particular (or makes everything look equally eye-catching) then it is failing to help the user to decide what to do. There are sometimes situations where it is useful or even vital to distract the user form what they are currently doing and encourage them to think about something else instead. These include situations where safety is at stake (such as the aircraft example described earlier), as well as more everyday systems.

In deciding what to emphasise and make more available, web page designers can often find out (from server access logs) which parts of a site are the most popular. The site can then be re-designed so as to make it easy for the user to focus on the most popular information and links.

Activity 6 - Getting the user to look in the right place

Study the description given by Bruce Tognazzini at his web site [http://www.asktog.com/columns/000maxscrns.html] of how progress was made on a particular, and simple sounding design problem. At each stage, consider where users were focusing their attention, and how what the designers did had an effect on this.

Review Question 4

What is attention? What are some of the consequences of failing to focus attention on the appropriate things in a user interface?

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

Review Question 5

What user interface design features affect users attention focusing? In what situations might the use of each be appropriate for in a user interface?

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

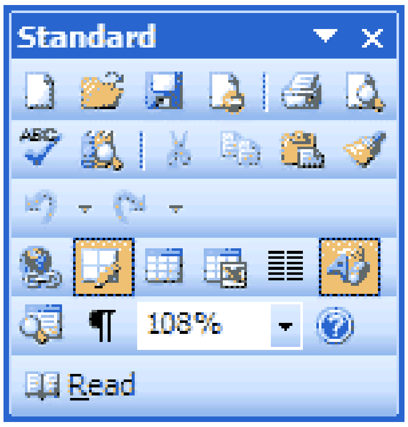
Memory

A further aspect of human psychology that is highly pertinent to the design of user interfaces is the study of memory: that is how we are able to store and remember information that is relevant to the computer systems we use and the tasks we use them for.

Every activity we engage in involves the use of our memories in one way or another. Not only is memory used in processing every input we receive from our senses, but also memory is essential for the fleeting, temporary or intermediate thoughts that are part of just about every task we carry out. At the opposite end of the spectrum, we continually need to remember facts we have learned previously about the world we live in – and the computer systems we use – in order to make decisions and carry out actions. Even when we are not aware that we are remembering things, we probably are! Skills that we acquire are often performed without conscious thought, but still require us to recall remembered abilities. Memory is just as central for an expert cyclist or a proficient typist, who may not be conscious of remembering the skills they have acquired, as it is for someone trying to recall facts during an examination, who may be painfully aware of their attempts to remember (or failures to do so).

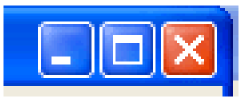
Failing to remember can result in a number of possible errors in carrying out tasks. The kind of "lapse" or error of forgetting mentioned in the attention section, which can lead to us omitting part of a task or carrying out the wrong task, is one example of this. An HCI-related example here would be failing to save a document before quitting a word processor, or failing to bookmark a web site before quitting the browser.

Another effect of failing to remember is when we simply cannot recall how to carry out a task at the time when we need to, or fail to recall some important fact about the workings of the computer system. An example is if we are unable to recall the meanings of visual elements of a user interface. See if you can you recall the meanings of each of the icons on this toolbar from a popular word processing program:



Some of the meanings are easy – the icon that looks like a printer invokes the print command – but others are more difficult – what is the meaning of the icon to the right of the printer?

A potentially more damaging instance of much the same phenomenon is if we mis-remember facts about the computer system or tasks we wish to perform with it. For example, if we incorrectly recall the meanings of the icons on the title bars of windows, we can end up quitting an application and losing work rather than simply minimising a window.

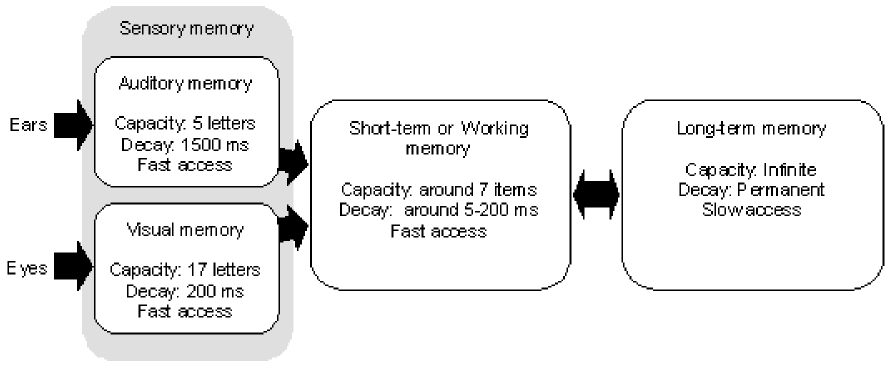


In the remainder of this section, we will look at how our memories work, and the three different kinds of memory store that studies have identified. We will then look at how what is known about memory can be used to improve user interface design.

The psychology of human memory

Psychological studies have revealed several types of memory system have a role to play in human cognition, each with its own specialised function and its own properties. Central properties that distinguish these types of memory are the capacity or size of the store, the decay time or the time for which items will persist before being forgotten, and the time it takes to recall or access items.

Three classes of memory that psychologists have studied extensively are sensory memory, short-term (or working) memory and long-term memory. The following diagram (extracted from the Model Human Processor of Card, Moran and Newell, 1983) summarises the characteristics of these memories and the relationships between them. Although the figure only shows sensory stores for the visual and acoustic senses, specialised memories exist for the other senses as well.



Sensory Memory

The first type of memory is responsible for recording information as it comes from our senses, prior to processing by other cognitive mechanisms. This so-called sensory memory acts as a kind of buffer store between the sensory organs and other forms of memory. In fact, the sensory memory is not a single entity, but a series of specialised stores, one for each of our senses. Thus images perceived by our eyes are retained briefly in the visual (or iconic) memory, and sounds perceived by our ears are stored in the acoustic (or echoic) memory. These memories store "images" of what is perceived for a short period of time, and can be thought of as being continually overwritten as new sense data arrives. In fact there are several different sensory memories: one for each of the senses, each having a different capacity and decay time, and storing images specific to a single sense.

Short-term or working memory

The second form of memory is referred to as either short term memory or working memory, and can be thought of as a kind of scratch pad in which information that is the current focus of our conscious attention is stored. The short-term memory typically stores sensory data from the sensory memories, and other short-term information that is needed as mental tasks are carried out.

Typically, our senses are bombarded by a vast amount of information about the external environment, and therefore a large volume of sensory stimuli enters the sensory memories. However, we choose to pay attention to only a small amount of this, and this attentional focus allows only a part of our sensory experience to be passed on to working memory.

In addition to processing sensory experiences, working memory also serves as a kind of "scratchpad" in which thoughts are stored while they are our focus of attention. For example, if you wish to multiply 39 by 8, it is necessary to remember these two numbers, multiply 9 by 8, to produce an intermediate result, remember the 30 and 8 and multiply them together, producing another intermediate result. The two intermediate results must then be remembered and added together, a process that involves further remembering of small amounts of information, which are stored for short periods of time, and then discarded.

Working memory has only a very limited capacity – studies have shown that it typically holds between 5 and 9 items (though what counts as an item is by no means fixed). Items persist in working memory for a relatively short time – from a matter of a few milliseconds up to tens of seconds). This means that working memory is a very limited resource, and if tasks make excessive demands on it, then errors are likely to result.

Activity 7 - Working memory

"Modes" are states of a system or user interface where the controls and inputs have the same effect. A user interface may have several modes, and the user's inputs will do something different in each, is which case the user must remember which mode they are in and what effect their actions will have. (For instance, Microsoft Word has Normal and Overwrite modes. In the former, characters that are typed are inserted in to the document, in the latter, existing text is overwritten by typed characters.)

Look at the user interface of your computer (if you want, select a particular application to look at), and find places where different modes exist, and where the user must remember the current mode. What can be done to reduce the errors that are likely to result from the presence of modes and the limits of working memory?

A Discussion on this activity can be found at the end of the chapter.

Long-term Memory

The third kind of memory is referred to as long-term memory, and is an effectively permanent and infinitely large store for semantically linked information.

A number of distinctions between different forms of long term memory have proved useful in the study of memory by psychologists. The subtleties of these different classes of long term memory, and the theories about how they function are not really relevant to this module, but it is worth noting that a variety of different kinds of information are stored in long-term memory. These include:

- facts about the world, computers and other devices we use, and tasks we carry out (sometimes referred to as semantic memory)
- things that have happened in the past (autobiographical memory)
- things to do in the future (prospective memory).

The processes by which we are able to remember items that are stored in long-term memory are complex and are the subject of much psychological research. However, a factor that greatly affects our ability to remember something is its familiarity, both in terms of its meaningfulness, and the frequency with which items have been remembered in the past.

This is why the use of graphical icons in a user interface can aid usability: the more meaningful a graphical element or a name appears to be, the more likely its meaning is to be remembered. For this reason, the meaning of menu items like "Print" and icons like

are more likely to be remembered than "Control-X" or "F4". In addition, in a given situation we are more likely to successfully recall a fact or piece of information that we have recalled often in the recent past, than something that is only occasionally required.

Note

Example: The Unix operating system.

In order to carry out any command, a Unix user must remember a great deal about the command, such as its name, how to supply parameters, the form of the output, and so on. Even to be able to use the online help system, the user needs to remember the command name. In some cases, the names bear a resemblance to the command they denote (the move command is "mv", the copy command is "cp", the print command is "lpr"). For some commands, the name is equally suggestive of the function, but the function is less frequently used and therefore harder to recall. In other cases, the connection between the name and its function is bizarre. Take, for instance the "biff" command, whose purpose is to notify the user when email arrives:

Did You Know?

"Biff was Heidi Stettner's dog, back when Heidi (and I and Bill Joy) were all grad students at U.C.Berkley and the early versions of BSD were being developed. Biff was popular among the residents of Evans Hall, and was known for barking at the mailman, hence the name of the command."

It is, of course, possible that if an amusing story is attached to a command, then a chain of associations between the command and the means of invoking it is produced, making it more memorable. But this is hardly a workable principle for the design of user interfaces!

Activity 8 - Long Term Memory and Interface Design

Investigate how much expert users know and can recall about a computer system they use routinely (obvious examples would be popular word processors, email programs, or web browsers). In particular, find out how much they know about the commands that are available, how the commands are named, and how they are accessed through the toolbars, menus and icons on the screen. A good way to do this would be to plan a structured interview or a questionnaire, to probe the user's knowledge of a selection of frequently used and less popular functions that the product supports.

A Discussion on this activity can be found at the end of the chapter.

Knowledge in the head and knowledge in the world

Memory research has shown that humans are far better at recognising material presented to them than they are at recalling material from memory.

One of the difficulties with text-based command-line user interfaces, as the Unix example above shows, is that the user interface affords very few opportunities for recognition, and relies almost totally on the user being able to recall the relevant commands from memory. Graphical user interfaces, on the other hand, make use of menus, icons, and similar devices to provide the user with on-screen representations of the commands that are available. The mental act of recalling how to carry out a particular function is therefore transformed into the rather less demanding task of recognising the visual representation of a command. In general, we are able to combine information we recognise in the environment with that recalled from memory in deciding how to act.

Experimental research in HCI suggests this is precisely the way people make use of menus and similar visual interface features. Even experienced users of a system have a very poor recollection of what menu items appear where in an interface, how they're labelled, and so on. What this means is that in order to be proficient at using a computer program, it is not necessary to remember precisely how the user interface works. It appears that instead, users decide what action to take largely on the basis of matching options and functions displayed on the screen (e.g., as menu items and headings) with their current goals and intentions. For example, if the user wishes to save the current document, they may well look for items that are relevant to this goal. Of the options that are available, the "File" menu tab will be recognised as the most relevant, leading the user to pull down that menu. Once the menu appears, a number of options appear, allowing the user a further opportunity for recognising the one that is most relevant: in this case "Print".

Donald Norman (see Norman, 1988/1990) has discussed the difference between recall and recognition in terms of the location of knowledge as being "in the head" or "in the world". What this means is that our actions are often governed by bringing together information that we recall from memory and information that is present in the external world. In everyday life, we frequently arrange our environment so as to place important information "in the world" where it is less susceptible to being forgotten or mis-remembered. Familiar examples of this are when we write notes to ourselves or lists of things to do at a later time, thus "off-loading" the responsibility for storing them.

As designers of interactive systems, it is possible to achieve similar effects, by placing important information about how a system works in the user interface, rather than relying on the users remembering it.

Implications for user interface design

From the discussion above, it should be clear that an understanding of how human memory works is very useful in designing human-computer interfaces. Some key points to bear in mind when designing user interfaces are:

- Reduce working memory load – try to minimise situations where the user must remember temporary information. For example, if the user needs to makes use of selections made earlier in a dialogue, then the choices made should be displayed so the user doesn't have to remember them.
- Recognition of interface elements presented to the user is far more effective that requiring the user to recall how to use a system
- Therefore, where possible put the knowledge about how a system should be used into the user interface.
- Graphical user interfaces typically require far less mental effort that text-based ones as they put more knowledge and information in the user interface. This allows the user to recognise what is in the interface, rather than having to recall facts from their memory.

- The use of meaningful and easy to understand icons, text labels, and menu items is a way to do this and therefore reduces the burden on the user's memory.
- User interfaces in which commands work in a consistent way reduce memory demands on the user. For instance, in Microsoft Word, the TAB key inserts a tab character – most of the time. When the cursor is inside a table, the TAB key moves the cursor on to the next box in the table! The designers have therefore forced the users to remember and recall an extra item from memory: the rule that governs how TAB behaves inside a table.

Review Question 6

Working memory is said to be limited in capacity to 7 +/- 2 items. What implications does this have for the design of menus in standard computer systems? Does it have implications for any kind of menu?

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

Review Question 7

What is the difference between knowledge in the head and knowledge in the world? Why is it important for user interface design?

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

Discussion Topics

People are good at adapting their environment in order to remember things. This is specially true for things they have to do in the future (prospective memory) and thing they have done in the past (autobiographical memory), but applies to other kinds of knowledge too. Examples are writing various kinds of notes (e.g., a list of things to do) and placing objects so that they are in view (e.g. placing letters to be posted near the door so that they're not forgotten). Some techniques are specific to certain jobs (such as aircraft pilots removing a glove as a reminder that a task is unfinished) or to certain cultures (such as tying a knot in a handkerchief as a reminder that something needs to be done).

How do you adapt your environment to help you remember things? How might the kinds of "external memories" you create provide inspiration for designing memory aids in user interfaces?

Answers and Discussions

Answer to Review Question 1

Sound is not persistent, carries relatively less information, conveys a single "linear" stream in information, tends to perceivable at a greater distance than visual output, is more or less non-directional, and it is difficult for us to choose to focus on only a part of our "audio field".

This means that sound is not necessarily appropriate conveying all types of information, and not good for conveying large volumes of information. But it is particularly good for providing certain kinds of alerts, giving short term feedback that actions have been carried out, augmenting interfaces to provide additional cues to the user (e.g., Earcons).

The non-directionality and range of sound may also have implications for privacy and use in situations where sounds would create a disturbance.

Answer to Review Question 2

Main advantage of voice is that it doesn't require a special phone.

The voice message may take longer to play than the visual presentation takes to read.

If the time at which the call was made is the important thing, the need to wait until the earlier part of the audio message has played might be frustrating.

Audio interfaces tend to place a greater burden on the user's memory.

Recording contains unnecessary words and could therefore be made more efficient.

Answer to Review Question 3

Time taken may be much greater than for sighted users.

Many web sites have substantial non-text content, such as graphics, animations and so forth, which a voice synthesiser would be unable to cope with. (The problem can be alleviated, but not eliminated if web page designers use ALT tags when images are included in web pages.)

Web pages are often designed in an inherently two-dimensional way, making use of layout and organisation in significant ways. A voice synthesiser, though, reduces all pages to a one-dimensional stream of words, losing much of the structure that is otherwise visually presented.

Answer to Review Question 4

Attention is the selection of one of a number of possible perceptual and mental activities to focus on. The idea is that we have limited cognitive capabilities, and therefore have to be selective in what we perceive or think about.

Inattention can result in failing to carry out tasks (forgetting them – a "lapse") or carrying out a similar, more familiar, task in place of the desired one (a "capture error"). Focusing attention of the wrong elements of a user interface can lead to distractions, making the user's task harder, and to the user missing important stimuli (such as notification of an emergency).

Answer to Review Question 5

Visual and audio alerts – to draw the user's attention to specific and important events and conditions (e.g., emergencies, or system generated warnings).

Motion, blinking, annoyance, etc – can be a distraction and an annoyance. Use only occasionally and with care!

Colour, intensity, size, etc. – to make specific pieces of information stand out from the background.

Spatial and temporal cues – to help the user to focus on where they are currently working in a large space (e.g., a form) or what stage they are at in a process with several stages (e.g., online purchasing).

Structure and layout – structuring can help users decide what to focus attention on when a large amount of information is presented, and to stay focused.

Answer to Review Question 6

Size limits of working memory have no implications for visual menu design - the user is not expected to retain all menu items in memory, that is what the menu labels are for!

However, working memory size and decay time is highly relevant for the design of audio menus, where the user may have to remember the menu items until after the menu has completely finished playing before making a selection.

Answer to Review Question7

Knowledge or information that is in the world (i.e., in the environment or the user interface) must be perceived and recognised by the user. Knowledge in the user's head must be recalled before it is of use.

People are generally better at recognising things than recalling material from memory. Therefore, if we can build more of the information about how to use a system into its user interface, we have reduced the demands on the user's ability to recall knowledge. Therefore the mental effort of using the system should be reduced.

Discussion on Activity 2

Wide / shallow schemes are unlikely to be as effective for audio menus. The benefit of having many options in a menu doesn't necessarily apply, since the user must wait for all of the options to be spoken.

However, one problem with deep audio menu structures occurs if user gets lost, makes an error, or has to backtrack.

One way to investigate this without the need to implement a voice menu system is to carry out a "Wizard of Oz" experiment. An experimenter plays the role of the computer, and speaks the menu items in response to the user's actions. See, for example, (Dix et al., 1998) or (Preece et al., 1994) for more details about this technique. The results, in terms of time taken, errors, and so on can be compared with results from using the menu structure of an application program.

Discussion on Activity 3

Applications are many and include selling anything where being able to smell, touch, or taste a sample of the product would make the user more confident that it is what they're looking for. For instance, users could smell food products, perfumes or cosmetics before they buy.

Touch could be similarly used to allow users to feel the texture of products before buying. Some sites marketing clothing attempt to give an idea of the texture of fabric used by providing pictures. This could be greatly enhanced if the user were able to feel "for real".

Discussion on Activity 4

You probably found that it takes longer to find information in the second display than the first. A study cited in Preece et al (1994) found that on average people took 3.2 seconds to find information in the first screen and 5.5 seconds in the second. Although the two screens contain roughly the same amount on information, the structuring of information in the first makes it easier for users to focus their attention on the particular type of information they are searching for.

Discussion on Activity 7

Many examples or modes exist. Already mentioned are is the TAB key which works differently in different places. Similarly, In Word, Control-D sometimes opens the Font dialogue box, and sometimes duplicates drawing objects; a title-bar icon with an X in it sometimes closes a document and sometimes an application plus documents; and so on.

Measures to reduce the problems include: reducing the number of modes and mode-dependent commands, making modes work in a more consistent way, making it very clear what the current mode is (so that the current mode can be perceived rather than having to be remembered). The best solution is to eliminate modes entirely and make commands work the same way in all circumstances, thus reducing demands on both memory and attention.

Discussion on Activity 8

Typically, expert users have a surprising poor recall of the functioning of the systems they use. What they do tend to have a better knowledge of is how to interpret information that is available on the screen, and well as information that is not available on the screen, See (Preece et al. 1994, Chapter 5) for a fuller discussion.

Chapter 6. Design Guides

Table of Contents

Context

In the previous two units we looked at aspects of cognitive psychology and how such knowledge can be used to build more effective computer systems. Researchers and practitioners have formalised the sorts of observation we made into various types of design guides. You will look at the different types of guide and the role they play. The main focus of the unit will be a review and discussion of principles than can be followed to improve a system's' learnability, flexibility and robustness.

Objectives

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

- Distinguish between design standards (e.g. house style and international standards).
- Discuss the role and application of design guides in interactive system development.

- Argue the case for good system learnability, flexibility, and robustness.
- Show how design principles can be used to improve learnability, flexibility, and robustness.

Introduction

The aim of this section is to discuss the role of standardisation in the domain of interface design. Standardisation can be viewed as a way of insuring that good human factors are incorporated in a system. In general, we are familiar with standardisation in many areas of our lives, for example, standard food labels, standard controls on cars, keyboards, colours for traffic lights and so on. The main purpose of standardisation is to provide safer and better quality.

For interactive system designers, designing systems with maximum usability is the ultimate goal. A good design mainly results from knowledge and experience of the designer and more importantly, the way they apply this knowledge into a specific design. In order to increase usability, two kinds of design rules, standards and guidelines, can be applied into a design.

Standards vs Guidelines

The difference between these is that standards are high in authority and limited in application, whereas design guidelines are low in authority and are more general in application.

The best user interface guidelines are high level and contain widely applicable design principles. The designer who intends to apply these principles should know which theoretical evidence supports them and apply the guidelines at an early stage of the design life cycle.

Standards	**Guidelines**
High Authority	Lower Authority
Little overlap	Conflicts, overlap, trade-offs
Limited application - e.g. display area	Less focused
Minimal interpretation-little knowledge required	Interpretation required -expert HCI knowledge

National and international bodies most commonly set interactive system design standards such as British Standards Institution (BSI), the International Organisation for Standardisation (ISO).

Design guidelines can be found in various forms, for example, journal articles, technical reports, general handbooks, and company house style guides. A good example to this is the guidelines produced by Smith and Mosier (1986) . An example of company house style guides is Apple's Human Interface Guidelines. Standardisation in interface design can provide a number of benefits.

Design Guides

Design rules, in the form of standards and guidelines, provide designers with directions for a good and quality design with the intention of increasing the usability of a system.

Software companies produce sets of guidelines for their own developers to follow. These collections are called house standards, sometimes referred to as house style guides. These are more important in industry, especially in large organisations where commonality can be vital. There are two main types of style guides: commercial style guides and corporate style guides.

Hardware and software manufacturers produce commercial style guides and corporate style guides are developed by companies for their own internal use. The advantage of using these guides is to enhance usability of a product through consistency. However, there are other reasons, for example, software developed for Microsoft, Apple Macintosh or IBM PC maintains its "look and feel;" across many product lines.

Though commercial guides often contain high level design principles, in most cases these documents are based on low level design rules which have been developed from high level design principles. If an organisation aims to develop their own corporate style guides for software, generally the starting point will be one of the commercial style guides.

Many design style questions have to be addressed more specifically when developing corporate style guides. For example, how should dialog windows to be organised? Which styles - colour and fonts etc, should be used ?

The roots for design rules may be psychological, cognitive, ergonomic, sociological or computational, but may, or may not be, supported by empirical evidence.

Design Principles

Dix et al (1998) split the design principles into three main categories.

Learnability

Learnability principles are concerned with interactive system features, which aid novice users to learn quickly and also allows steady progression to expertise. The principles discussed below support the learnability design principle.

Predictability

This interactive design principle requires a user's knowledge of interaction to be sufficient to determine the outcome of present or future interaction with the system.

One form of the predictability principle is concerned with a user's ability to envisage which operations can be performed next. This is often referred to as operation visibility, and is concerned with showing the availability of operations which can be performed next by the user. Based on this principle, if an operation can be performed then there should be a clear indication of this to the user.

For example, closing a document should always allow the user to save changes not saved already.

Synthesisability

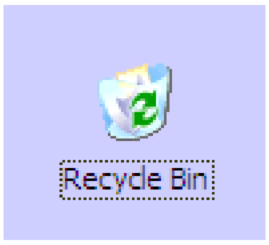
Two aspects of synthesisability are immediate honesty and eventual honesty. In general, these principles relate to the ability of the interactive system to provide the user with an observable and informative notification about the operation state changes within the system.

A good example of this is the file management capabilities of Windows Explorer and the command line operations in DOS. Moving a file from one folder to another is observable by the user in Windows, however, carrying out the same operation in DOS provides no visual representation of the system's actions, in other words no immediate honesty.

The problem with eventual honesty is that if the user is a novice user, not familiar with the system's operations, synthesising the consequences of the operations carried out by the system may be more difficult.

Familiarity

The familiarity principle is concerned with the ability of an interactive system to allow a user to map prior experiences, either real world or gained from interaction with other systems, onto the features of a new system.



A recycle bin is a familiar item in the real world and recycle bin icon immediately suggests its function.

This type of familiarity is also referred to as the guessability of features in the system. Another example of this is the similarity between the computer keyboard and that of a typewriter.

The appearance of an object provides familiarity with its behaviour. Effective use of appearance in an interactive system design can improve the familiarity of the system.

Generalisability

This interactive design principle provides support for users to extend knowledge of specific interaction within, and across applications, to new, but similar situations. For example, cut/copy/paste operations within Microsoft Office applications use of same short-cut keys.

Similarly if a user knows how to draw a rectangle using a drawing package then the user can apply this knowledge to draw a circle using either the same package or other similar packages.

One of the purposes of standards, and programming style guides, is to increase generalisability across a variety of applications.

Consistency

To support generalisability, consistency is essential and is probably one of the most widely applied design principle in user interface design. Consistency between application is always favourable, however consistency within an application is essential.

Standard GUI design factors should aid designers to take into account consistency at every level, and, "look and feel" issues should never be abandoned. The use of labels and icons should always be consistent and the same icons and labels should mean the same thing. The principle of "sameness" should be applied to the use of terminology, formatting and input/output behaviour arising from similar situations or task objectives.

Flexibility

Flexibility in interactive design extends the way a user and the system exchange information. By applying flexibility principles to an interactive system design, designers aim to improve a system's usability.

Dialog initiative

When the system controls the dialog flow, the dialog is said to be system preemptive. Conversely, when the flow is controlled by the user, the dialog is said to be user preemptive. In general a user preemptive dialog is favoured although some situations require a system preemptive dialog. In reality some line between these two extremes is usually the most satisfactory solution.

Multi-threading

Within a user interface a thread can be considered a part of dialog that allowing a task to be performed. Multi-threading within a interface provides support for multiple tasks to be performed at one time.

Multi-threading can be described as concurrent or interleaved. An interleaved system permits work on a single task at a given time - a word processor allow multiple documents to be open, but only one can be worked on at any instant.

A concurrent system allow multiple tasks to be actioned at a given time - within Windows a document can be edited in MS Word and while the file find utility is active.

Task migratability

Task migratability means passing responsibility of execution of tasks between user and system. A computerised spell checker is a good example to this. It is a waste of time for a user to manually check a very long document and correct. A spell checking facility in a word processing application can check words against its own computerised dictionary. However, it is considered 'dangerous' to allow a spell-checker program to carry out this task without the user's assistance.

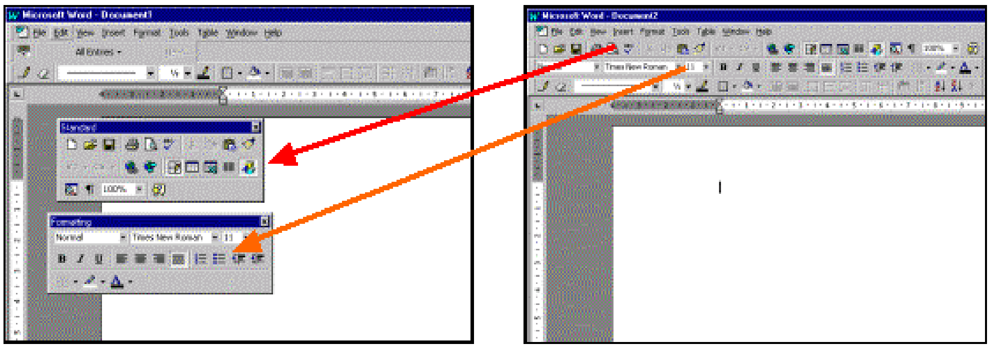
Substitutivity

Substitutivity offers a user alternative ways of specifying input or viewing output. Indeed the distinction between output and input can be blurred. For example, a drawing package may allow start and end co-ordinates of a line to be specified, conversely, the same system may allow the line to be drawn first, and the system indicates the end point co-ordinates.

Customisability

The user interface should be able to support individual preferences. For example standard control bars in MS Word can be amended as required.

The customisability principle supports a user's ability to adjust systems settings or features to a form that best suites the preferred way of usage.



Adaptivity can be automated but in order to be able to provide such user-centred system behaviours the system should be trained to distinguish an expert's behaviour from a novice user's behaviour. Repetitive tasks can be detected by observing a user's behaviour and macros can be automatically constructed.

Robustness

The robustness of an interface design can be measured in terms of the following four principles. These principles aim to support users to achieve their goals.

Observability

Observability should provide users with an ability to evaluate the internal state from its representation. If a user cannot understand the internal state of the system, there is a high likelihood that the user's confidence will be very low, for example, if the system is performing a time consuming operation, the current status of the operation should be displayed - a web browser will indicate the on-going status of a page download.

There are several aspects to system observability. You should read up more on this topic in your textbook and/or on the Internet.

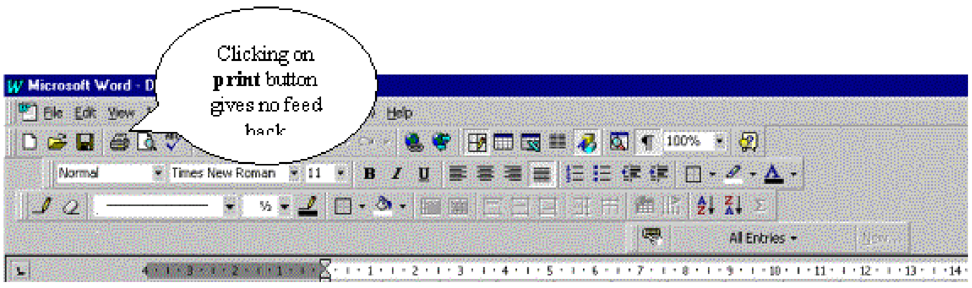
Recoverability

Users should be able to reach a desired goal after recognition of errors in previous interaction. Error recovery can be achieved in two ways, forward (negotiation) and backward (undo). Forward error recovery involves a user accepting the current state of the system and negotiating from the present state towards the required state. A backward error recovery mechanism within a system allows a user to undo the undesired outcome of the previous interaction by returning to a previous state.

In addition to providing the ability to recover forward or backward, the effort to achieve this should reflect the work being done - the commensurate effort.

Responsiveness

Responsiveness is usually measured in terms of the rate of communication between the system and a user. Response time, indicating change of states within the system, is important. Short duration or instantaneous response time is more desirable.



When an instantaneous response cannot be given by the system, the system should be able to indicate to the user that the system has received the request and in processing an appropriate action (see definition of observability).

As illustrated in the above picture, clicking the print icon on the Microsoft Word menu does not give feedback, e.g. especially novice users may end up pressing the print icon a number of times.

As well as response time of a system, consistency, of responsiveness is also desirable.

Task conformance

There are two aspects of task conformance, task completeness, and task adequacy. Task completeness is concerned with whether a system is capable of supporting the entire task that a user wishes to perform. The task adequacy is concerned with addressing the user's understanding of these tasks It is necessary that an interactive system should allow the user to perform the desired tasks as defined during the task analysis.

Activities & Review Questions

Activity 1 - Consistency for data display interface design

Consider the following menu/actions available within a system.

Case 1	**Case 2**	**Case 3**
delete/insert character	remove/insert character	delete/insert character

Case 1	**Case 2**	**Case 3**
delete/insert word	delete/insert word	kill/insert word
delete/insert line	create/insert line	delete/insert line
delete/insert paragraph	kill/create new paragraph	delete/insert paragraph

A good design principle is to create a consistent and familiar user interface. Comment on the above design.

A Discussion on this activity can be found at the end of the chapter.

Activity 2 - Importance of producing standards for interface design

Why is producing standards for interface design important? What kind of benefits can they provide?

A Discussion on this activity can be found at the end of the chapter.

Activity 3 - Design principles and rules

Which of the following can be classified as principles and which of the following can be classified as design rules?

1. Always prompt a 'warning' message to the user before deleting a file.
2. Provide a 'SAVE' command.
3. Reduce cognitive load.
4. Ensure consistency in presentation.
5. Provide a 'HELP' facility.
6. Ensure efficient use.
7. Ensure operational visibility.
8. Display an 'EXIT' command always at the bottom of the 'FILE' menu option .
9. Design for user growth.

A Discussion on this activity can be found at the end of the chapter.

Activity 4 - House style guides - corporate style guides

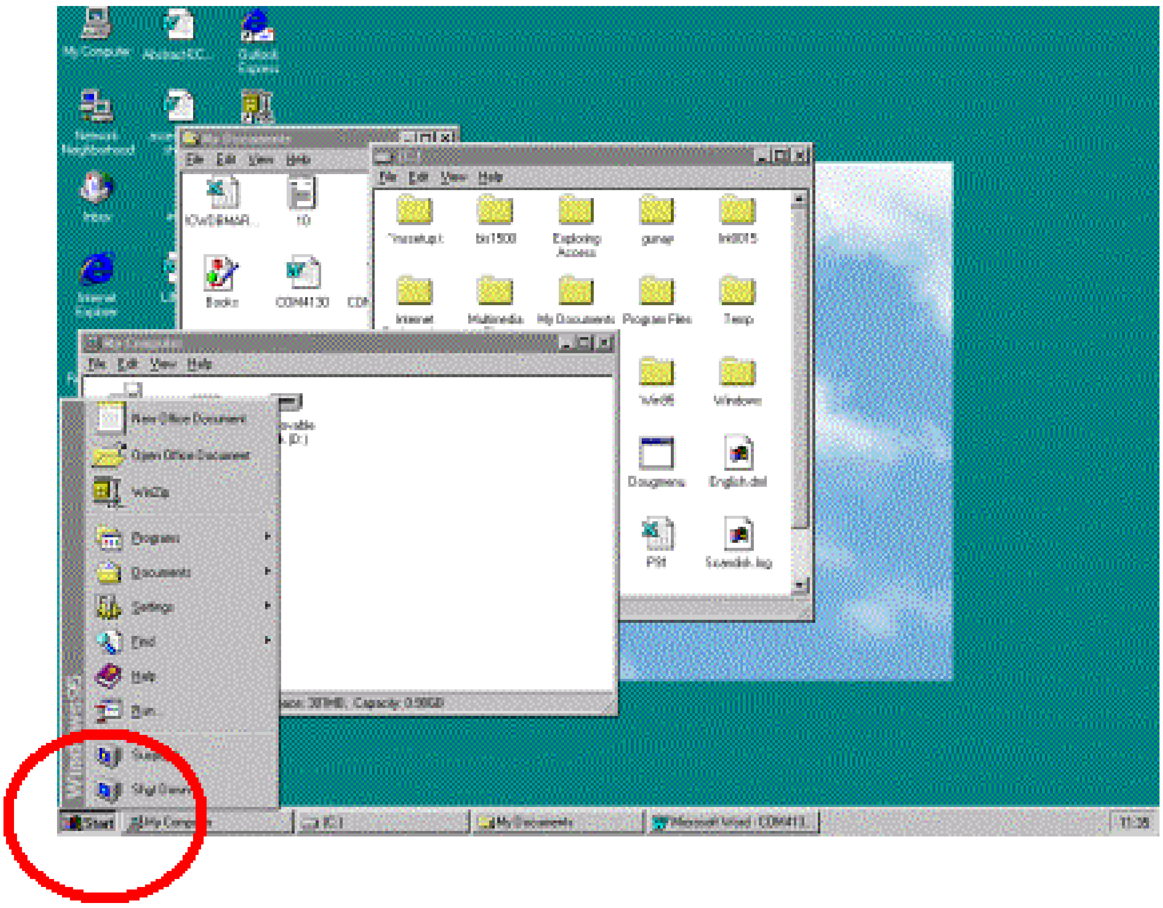
Sketch the screen design of a system to achieve the following. On receiving a house number and postcode from a user, the system must display the location of the address on a map of London. Provide a suitable mechanism to receive input from a user and report errors.

Assume that your design will be implemented using a PC-style interface. Detailed designs of interface items such as buttons and error dialog boxes should be determined at this stage of the design.

A Discussion on this activity can be found at the end of the chapter.

Activity 5 -User interface design principles

Since Windows 95, Microsoft's desktop operating systems' shut down function is accessed via the start menu. Which of the design principles can this be said to contravene?



A Discussion on this activity can be found at the end of the chapter.

Activity 6 - User preemptive dialog

Why is a user preemptive dialog generally preferable? Why is it not always practical?

A Discussion on this activity can be found at the end of the chapter.

Activity 7 - Customisability

What are the advantages of providing a customisable interface? What are the possible dangers?

A Discussion on this activity can be found at the end of the chapter.

Review Questions

1. What advantage could a large software company gain in constraining the creativity of its designers?

   Answer at the end of the chapter.

2. Would you classify the following guidelines as design principles or design rules?

   - Easy to use
   - Adapted to users' level of knowledge
   - System to provide feedback to user

   Answer at the end of the chapter.

3. What problems can an inconsistent interface cause to users?

   Answer at the end of the chapter.

4. What are most design guidelines based on?

   Answer at the end of the chapter.

5. Why is it essential to provide observable feedback when performing an action in an interactive system?

   Answer at the end of the chapter.

6. With reference to task migratability, why may it be dangerous to assign too much control to a system?

   Answer at the end of the chapter.

7. List the three categories of design principles which support the usability of interactive systems.

   Answer at the end of the chapter.

Discussion Topics

Human interface guidelines are often based on experiments. More experimentation could lead to refined standards and defendable. Since we will technology changes very rapidly, we may never have a complete set of guidelines.

Answers and Discussions

Discussion of Activity 1

Although unintuitive the actions for case 1 are consistent. In case 2, there is an inconsistency as an insert operation is present under 'delete' and 'create'. In case 3 a number of characteristics are inherited from case 1, however, access to insert functionality is not consistent. Although no option is ideal, on balance, case 1 would be preferred, case 2 is the least desirable.

A consistent interface is one where similar interactive situations are presented in a similar fashion and exhibits similar behaviours.

Consistency provides a firm foundation for a dialogue to proceed - if a designer ignores the principle of consistency, the user's ability to learn the system and operate effectively will be reduced dramatically.

Discussion of Activity 2

You should read up more on this topic, but the following table may serve as a starting point.

Common terminology	Standard measures of usability or performance mean that designers and user know that they are discussing the same concept.
Maintainability and evolveability	Standard implementation techniques facilitate program maintenance, because all programs can be expected to have a shared style and structure.
Common identity	House or industry standards for display style or screen layout ensure that all systems have the same "look and feel and are easily recognisable.

Less training	Knowledge can be transferred more easily from one system to another if standard command keys and other interaction techniques are adopted by interactive system designers.
Health and safety	Users are less likely to be surprised by unexpected system behaviour if standard controls and warnings are used. Unfriendly systems can be a source of stress.

Discussion of Activity 3

Five of these can be classified as design principles: 3, 4, 6, 7 and 9. The rest 1, 2, 5 and 8 can be considered design rules.

Discussion of Activity 4

Three main elements of style which need to be considered when developing a corporate style guide are:

1. Which style is to be used?
2. How should colour, graphics and other overlays be applied?
3. Which interface components should be used and for which purpose , e.g. when should a button be used, when should a check box be used and so on?

Discussion of Activity 5

Two learnability design principles, familiarity and predictability are contravened. One would not immediately associate the shut down operation with the start button and without prior knowledge one would not have any indication how to shut down the system. However, this is a minor criticism as once the route to the shut down option is learned it is easily accessed again in the future.

Discussion of Activity 6

People have different ways of working A system preemptive approach may enforce a way of working that is not natural for all users. However, a system preemptive approach may be necessary to control certain processes such as a logon procedure before allowing access to an application. It would not be sensible to allow access to a system before verifying logon details. A system-guided approach may also preferred when a task is performed for the first time or for rarely performed tasks

Discussion of Activity 7

New and novice users of an application are likely to initially need support when using a system. As they become more proficient they may develop specialised requirements or find certain features more useful, initial configuration cumbersome.

A customisable interface can be adapted to cater for on going and changing requirements. The danger of customisability is that a system may not be configured adequately for a particular user or a user could be presented with several configurations at different points in time, which may be confusing.

Answer to Review Question 1

First reason is to ensure consistency across all product lines within the organisation and the second reason might be to speed up the design process; if fewer design choices are open to designers, it should take less time to produce a good design.

Answer to Review Question 2

These guidelines are high level principles because they require further interpretation before being applied. (You may wish to refer to Activity 3).

Answer to Review Question 3

Mistakes (some of which may be safety-critical), cognitive overload and, stress.

Answer to Review Question 4

Guidelines may be based on psychological theory, however many are formulated from experience.

Answer to Review Question 5

If there is no observable effect when an action is performed it will not be apparent that anything has happened. If an action cannot be performed a clear explanation as to why should be provided and the status of a long running task should also be displayed.

Answer to Review Question 6

Not withstanding the advances in Artificial Intelligence, computer systems do not always handle situations sensibly. It is thus advisable to offer an option to allow system made changes to be reviewed and deemed appropriate, undone.

Answer to Review Question 7

Learnability, Flexibility and Robustness

Chapter 7. Models of the User

Table of Contents

Context

Whenever something is built – be it a bridge, handheld Web browser or a new workflow application – models are likely to be involved. Models are used both to evaluate a design (have we made the right choices?) and in the construction process itself (i.e. helping us to generate designs and other artefacts). Models are important in interactive systems design and evaluation. In this unit you will look models and modelling techniques that focus on users. The unit begins by looking at user requirement modelling and continues by considering two example cognitive models (GOMs and KLM) that are used to capture and evaluate user information processing behaviour.

Review Question 1

Why are models important in the design process? Consider what advantages there are to using models.

In what ways do models contribute to the design process?

Answer at the end of the chapter.

Review Question 2

Both user centred requirements analysis and cognitive modelling employ models. Consider what the fundamental differences between the two approaches use models.

Answer at the end of the chapter.

Objectives

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

- Understand the key concepts concerned with modelling.
- Explain why and how models are used in interactive systems design.
- Explain why user-centred requirements modelling is more effective for interaction design than traditional methods.
- Describe several user-centred techniques that can be used to capture user requirements (e.g., Soft Systems Methodology & participatory design).
- Explain what is meant by a cognitive user model, give an overview of the range of models available and indicate the usefulness of such models
- Distinguish between the high-level (GOMs) and low-level (KLM) cognitive models.
- Produce GOMs and KLM analyses of simple user tasks.
- Show how GOMs and KLM can be used to analyse an existing system or proposed design.

Models of User Requirements and Context

There are typically several interested parties in the design and construction of systems. These include the clients who commission the development, the designers and developers who design and build the system, and their management who look after the running of the project. These people are primarily interested in what the system should do – its functional requirements. An important set of people who are often forgotten about are the users who will actually end up using the system. In order to build systems which users will find usable we need to consider their needs rather than exclusively concentrating on functional requirements of the system. These non-functional requirements include issues such as the usability of the system and how acceptable it is to the users. Without such considerations systems may be produced which have been designed without considering whether they would be acceptable to users e.g. the introduction of an automated ordering system at a restaurant may not be acceptable to the users (waiters) if it requires them to spend a large proportion of their time interacting with it rather than interacting with restaurant goers. Clearly, if users do not find a system acceptable they will be reluctant to use it and so extensive redesign or user retraining may be necessary costing large amounts of time and money.

The rest of this section looks at approaches which are considered user centred. That is, they focus their attention on the user and their non-functional requirements rather than concentrating solely on functional requirements.

Socio-technical Models

Socio-technical models consider the users within the organisational context that the system will be introduced into. The particular focus is on the interplay between the social and technical issues - hence the term socio-technical. One such approach is CUSTOM which concentrates on identifying who will be involved with the new system, and what their requirements are (not just functional requirements). Once these have been identified the designers of the system should have a better grasp of the non-functional user requirements as well as the organisational structure itself and how this relates to the design. Taking these requirements into account should help designers develop more acceptable and efficient systems.

CUSTOM refers to the people who would in some way be involved with the new system as stakeholders who can be classified in four ways:

- **Primary** – the people who will use the system e.g. the waiters in a restaurant.
- **Secondary** – people who produce input for the system, or receive output from the system, but do not directly use it e.g. the restaurant goers who are presented with a bill produced by the system at the end of their meal.
- **Tertiary** – people who are touched by the success or failure of the system, but are neither primary nor secondary stakeholders e.g. the owner of the restaurant chain.
- **Facilitating** – the people involved in the system's design, development, and maintenance.

Once the stakeholders have been identified their characteristics are analysed to develop user centred requirements for the system. Dix, in his 1998 book, summarises this process of stakeholder analysis in terms of the following questions:

- What does the stakeholder have to achieve, and how is success measured? E.g. waiters have to ensure diners are served at appropriate times and are happy with the level of service (not too intense or too disinterested). One way to measure a waiter's success may be the size of their tip.
- What are the stakeholder's sources of job satisfaction? What are the sources of dissatisfaction and stress? E.g. for a waiter this may be the pleasure of serving food and providing a pleasant eating atmosphere. They may be stressed by angry customers or a large number of customers to keep happy at the same time.
- What knowledge and skills does the stakeholder have? E.g. a chef has extensive knowledge of cooking which the waiters may not.
- What is the stakeholder's attitude towards work and computer technology? E.g. the owner of a chain of restaurants may be a technophile whilst a chef may be a technophobe. This may well cause conflict in the introduction of new technology.
- Are there any work-group attributes that will affect the acceptability of the product to the stakeholder? E.g. is there something about people who become waiters that will affect how well they accept the product?
- What are the characteristics of the stakeholder's task in terms of frequency, fragmentation, and choice of actions? E.g. a busy waiter will typically have to perform many fragmented tasks with high frequency in order to keep the diners happy.
- Does the stakeholder have to consider any particular issues relating to responsibility, security, or privacy? E.g. waiters may need to be discreet with regular diners who dine each night with a different partner, and may need to ensure that credit card payments are dealt with securely.
- What are the typical conditions in which the stakeholder is working? E.g. the chef of the restaurant typically works in a hot and dangerous environment whereas the owner of the chain of restaurants may work in a conventional office environment.

Note how little is asked about the technology and the functional requirements of the new system. The intention is to understand users' working context and their possible attitudes towards the new system so that it will hopefully be more acceptable.

Activity 1 - CUSTOM

A bank has asked a systems development company "Cash Machines R Us" to introduce new cash machines into one of their branches. Use the CUSTOM approach to socio-technical modeling to identify who the stakeholders are in the introduction of a new cash machine.

A Discussion on this activity can be found at the end of the chapter.

Soft-Systems Methodology

Soft-Systems Methodology (SSM) takes a broader view than socio-technical approaches by considering the organisation as a whole. From this perspective the stakeholders and technology are components of the larger context. As with other user centred approaches, the intention is to understand the context of the people involved with the system rather than making explicit design decisions. These models of the situation are instead used to inform design and help designers understand the context in which the final system will fit.

SSM involves the development of three kinds of models to help understanding the organisational context. The first kind of model is referred to as the rich picture which provides a detailed description of the problem situation. This includes details similar to the models of stakeholders developed in socio-technical approaches discussed previously – who the stakeholders are, what groups they work in, and what tasks they perform. In addition the organisational structure is modelled where it impacts on stakeholders' work. As this model provides the context for further models it should be clear and informative for designers. It could be developed from interviews with people in the organisation, observations of their work practices, or more interactive approaches such as workshops in which stakeholders act out situations in order to help explain their work context.

The next stage moves the focus of analysis from the real-world situation to the development of definitions of what stakeholders perceive to be the activities taking place in the organisation. These definitions are referred to as root definitions of the system. There may be several different root definitions – representing different stakeholders' perspectives – which need to be reconciled at a later stage. Root definitions model the following aspects:

- **Clients** – people who benefit or accept output from the system e.g. in our restaurant example the clients may be the diners who benefit from the restaurant nutritionally and receive output from the system by way of a bill.
- **Actors** – stakeholders who perform activities in the system e.g. the waiters and chefs in the restaurant.
- **Transformations** – what changes the system performs on things in the environment e.g. a system which produces bills in a restaurant transforms diners' requests for food (conveyed by the waiters) into bills by the end of the meal.
- **World view** – how the system is perceived by a client e.g. a waiter may perceive an automated billing system as beneficial as it reduces the work they have to do in maintaining the bills for multiple diners.
- **Owner** – who the system belongs to, and who can allow changes in the system e.g. the owner of the restaurant chain owns the automated billing system.
- **Environment** – what factors influence the system e.g. a restaurant has to abide by certain health and safety standards.

Finally a conceptual model is constructed which details what the system has to do to meet the root definitions. The most important part of the root definitions are the transformations which are used in the conceptual model to define what is achieved, and how it is achieved. These achievements are usually modelled hierarchically to provide different levels of detail – much as tasks are decomposed in task analysis (see Unit 8). For example, the overall achievement of serving a diner includes successfully finding out what the diner wants, serving them, clearing the table, and ensuring that the food is paid for. Considering the hierarchic structure, achieving payment for the food involves several sub-achievements such as producing the bill, collecting the money, and possibly producing a receipt.

The conceptual model is used to identify differences between the real-world situation and the model of how the stakeholders perceive the system. These differences can then be used to inform change and/

or development of appropriate systems. The key outcome of the whole SSM approach is for designers to have a better understanding of the context in which developed systems are to be placed.

Activity 2 – Soft Systems Methodology

For the example in Activity 1 construct a root definition for the new cash machine from the perspective of a bank customer who wants to withdraw money.

A Discussion on this activity can be found at the end of the chapter.

Review Question 3

Discuss the similarities (and differences) between socio-technic and soft systems methodology approaches to understanding user requirements , what sorts of things do they look at, and how can this provide input into design?

Answer at the end of the chapter.

Participatory Design

The two approaches we have looked at so far develop user centred models which are typically used in the early stage of system development. In contrast, participatory design aims to keep the whole process of developing a system user centred. This is achieved by including users in the design team rather than treating them as subjects of analysis who remain outside the core design situation. The motivation for this is that users are essentially experts on their work situation – they know full well what work they do, how the organisation works for them, and who they need to work with – and so including them in the design process is essential if the new design is to fit well into their context. Moreover, the introduction of new systems typically changes the work context. For a new system to be acceptable these changes in the work context also need to be acceptable. Having users in the design team should help ensure that these changes do not overly disturb other users.

Participatory design has three main characteristics:

1. **Work focused** – design concentrates on improving the workers' environment and tasks they perform rather than focusing on the system requirements.

2. **Collaboration** – the designers and users collaborate on the design so that the users can contribute at every stage.

3. **Iterative** – design does not just happen once, the emphasis of participatory design is on several design and evaluation stages which build to a final design.

As participatory design focuses on the collaboration between users and designers it needs to employ various techniques and models to communicate ideas between them including:

- **Brainstorming** – sessions in which users and designers generate a range of ideas which are developed without judgement. These ideas are then fed into other techniques to be developed further or dropped.

- **Storyboarding** – a rough idea of a user's activities can be presented via a storyboard. The storyboard models user activities as a series of drawings – a little like a comic strip description of the activity. They are simple to develop and help users communicate with the designers about what they do, and how they do it.

- **Workshops** – these provide a forum for discussion in which designers and users can ask each other about their perspectives and so establish common understandings of the design issues as well as focusing their views of the design. Typically they are used to fill in gaps in understanding about the situation. As such the designers usually enquire about users' work environment, and the users ask about technological possibilities.

- **Pencil and paper activities** – this provides a more interactive way of talking through designs than storyboards. Typically, rough designs are drawn (i.e. possible designs are modeled using pencil and paper) and then users consider how they would use it by moving through the design step by step. Problems with the design can then be identified from trouble the users have as they move through it. These problems can then be addressed in the next iteration of the design.

Participatory design was developed in Scandinavia where it has been widely accepted. However, the time and effort it requires, and reliance of less hierarchically structured organisations (where designers and users are treated equally), has meant that its use is less wide spread outside Scandinavia.

Activity 3 – Participatory Design

For the example in Activity 1 construct a storyboard of a customer withdrawing money from a cash machine. Imagine that the cash machine has several pages of information which it displays. For any operation the first page allows the user to enter their PIN (Personal Identification Number). If this is correct for the customer's card the machine shows the next page which allows the user to select one of several services on offer. If they select the withdraw cash service they are presented with a page from which they can select a predetermined amount of cash, or can select an option to allow them to determine how much they want. If they select this option they are then presented with a page which allows them to enter a value up to £250. Once the amount of money has been entered (either by selecting a predetermined amount or entering their own amount) the machine returns the card and then the cash to the customer.

A Discussion on this activity can be found at the end of the chapter.

Review Question 4

How does participatory design relate to approaches such as socio-technical and Soft Systems Methodology. Concentrate on what they aim to do, and what sort of models they use.

Answer at the end of the chapter.

Summary of User Requirements Modeling

This section looked at models of user requirements. These are representations of the users' world which are used to inform the understanding of non-functional requirements such as whether the developed system would be acceptable in the workplace. We looked at socio-technical models which consider who the stakeholders in the development of a system are. We then considered soft systems methodology which takes a broader view as it also models the organisation in which the stakeholders work. Finally we touched on participatory design which includes users in the design process and uses models to help designers and users understand each others' points of view. All of these approaches involve extra time and effort than simply defining the functional requirements of a system. However, systems need to be developed which take into account users' needs and work context otherwise they may be unusable or unacceptable to the intended users. It is the effectiveness of user centred approaches to take this into account that makes them attractive to developers.

Cognitive Modeling

As we discussed at the start of this unit, a model is a representation of something that we can easily work with. In the previous section we looked at models of user requirements – essentially models of users' context. In contrast, this section considers models of users' cognitive abilities. That is, rather than looking at the context in which users work we will consider models of how people behave. These cognitive models are important because they provide a way of understanding people's cognitive abilities which can be used to inform design and, as we shall see in Unit 9, to evaluate systems. In this section we look at two cognitive models: GOMS and ICS.

GOMS

GOMS uses an explicit model of cognitive processes developed from studies of users. GOMS itself stands for Goals, Operators, Methods, and Selection rules. This core set of concepts describes the tasks that users perform, and how they achieve them. GOMS is more than just a way to describe tasks though, it provides a family of modelling techniques which can be used to generate predictions of the time to complete tasks given descriptions of the tasks and the user interfaces to be used. These predictions are based on a simple model of cognition called the Model Human Processor (MHP) and have an absolute accuracy of between 10% and 20%. The basic assumption of GOMS is evident in its name – that users understand their goals, GOMS considers the actions to meet such goals. Having made this assumption GOMS can give us feedback on the coverage and consistency of the user interface. By coverage we mean whether the user interface contains the necessary functionality to support the tasks it was designed for. Consistency refers to whether similar tasks are performed similarly with the user interface. Moreover, GOMS can give an indication of whether frequent goals can be achieved quickly using the given interface.

Model Human Processor

The central part of GOMS (Card, Moran, and Newell, 1983; see Chapter 6 of Dix 1998 for a summary) is the Model Human Processor (MHP). This is a simplified view of the psychological processes involved in human cognition which can be used to make approximate predictions of user behaviour. Card et al. based this model on empirical evidence i.e. psychological studies of users. Even though it is based on psychological evidence and theory the motivation for the MHP is that is should be usable by non-psychologists e.g. designers and evaluators. The MHP is consists of:

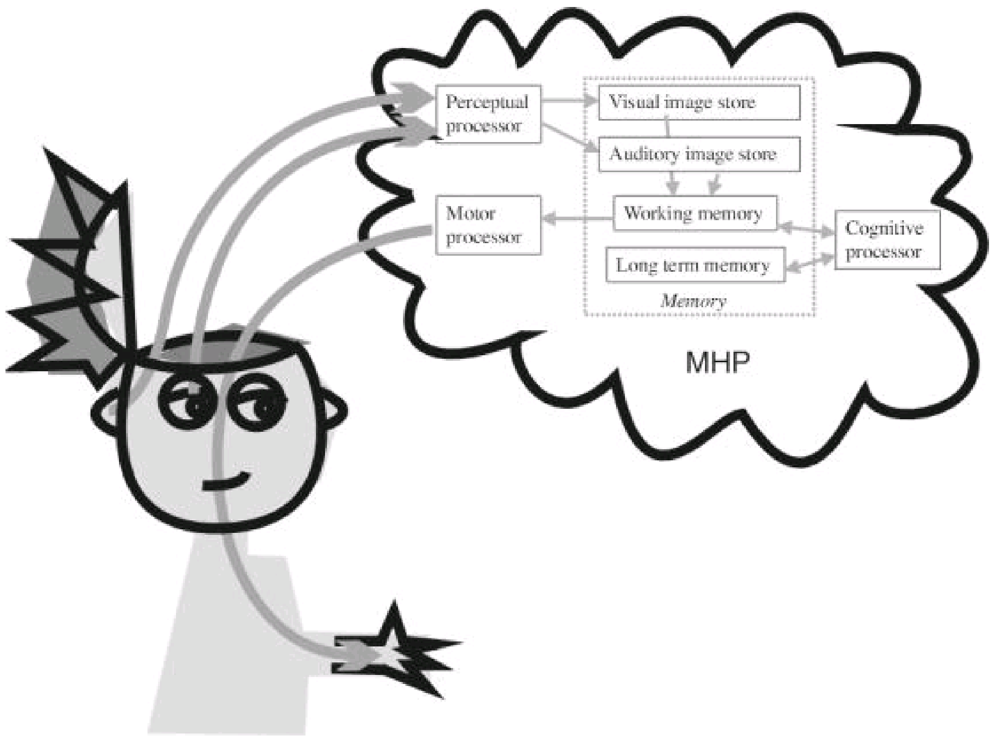
- **set of processors** – systems which process information and may make decisions
- **set of memories** – areas in which information is stored in various forms by processors
- **set of principles of operation** – these guide the operation of the processors and are crucial to a realistic account of human behaviour

The MHP itself is divided into 3 interacting subsystems: the perceptual system, the motor system, and the cognitive system. Each of these subsystems has their own memories and processes and works on different kinds of information as illustrated in the following diagram.

1. **Perceptual system** - input from the eyes and ears are stored symbolically in visual and auditory image stores which go on to be processed in working memory by the cognitive system, and possibly stored in long term memory for later access. E.g. the sound of a car horn is coded as a loud warning.

2. **Cognitive system** - processes information from working and long term memory to make decisions about how to respond. E.g. processing the fact that there is a loud warning noise, recalling that evasive action should be taken in such situations, and deciding to run to the pavement.

3. **Motor system** - executes responses to decisions made by the cognitive system. E.g. running to the pavement.



Each component of these systems has parameters which are used in the generation of behavioural predictions. Each memory system is defined in terms of its storage capacity (how many pieces of information can it hold at one time), and the decay time of an item (how long is it before an item is forgotten). Similarly, each processor is defined in terms of processor cycle times – how long it takes to process one piece of information. Card et al. (1985; see Johnson, 1992, for a summary) defined approximate values for these parameters based on psychological research which are listed below. Note that these values assume skilled (errorless) performance and only predict time for responses to stimuli – not for self initiated actions.

Table 7.1. Memory parameters

Memory type	**Storage capacity**	**Decay time**
Auditory	4 letters	1500 msec
Visual	17 letters	200 msec
Working memory	7 chunks of information	7 sec
Long term memory	Unlimited	Never

Table 7.2. Processor parameters

Processor type	**Processing cycle time**
Processor type	100 msec
(eye movement)	230 msec
Cognitive	70 msec
Motor	70 msec

So, we have described the MHP, but we need to return to the core of GOMS – the goals, operators etc. – before we can see how these parameters are used to make predictions.

Goals

Goals are something that the user wants to achieve e.g. go to airport, delete a file, or create a directory. They have a hierarchical structure – that is they are composed of many sub-goals which need to be achieved to meet the larger goal. These are similar to the goals identified in Task Analyses (see Unit 8).

Operators

Operators are elementary (can not be decomposed into smaller operations) perceptual, motor or cognitive acts which are necessary to change user's mental state or environment. As such they are the lowest level of a GOMS analysis. Using GOMS a user's behaviour can be recorded as a sequence of operators as operators can't occur concurrently. They are a similar level of description as actions in task analysis (see Unit 8).

For example, to move a file to a different folder the user might perform the following operations:

- Move cursor to item
- Hold mouse button down
- Locate destination icon
- Let go of mouse button

Methods

From operators we build up methods which are sequences of steps that accomplish a goal (and so are like tasks in a task analysis (see Unit 8)). As with goals these methods can include other (sub) goals. A fundamental assumption in GOMS is that methods are learned and routine (so no problem solving involved), and that there is only one way a user stores knowledge of a task.

For example, a user moving a file to a different folder could be described in GOMS as:

- **Goal** – move file to a different folder
- **Method** – move file
- **Operators** - Move cursor to item, Hold mouse button down, Locate destination icon, Let go of mouse button

Selection Rules

If there is more than one method to accomplish a goal, the Selection rules tell you which method to use. Again, as with methods, they assume error-free performance (so the user does not selected the wrong method by accident). They are written as IF … THEN statements as below:

```
IF <condition> THEN accomplish <GOAL>

For example:

IF <restaurant accepts credit cards> THEN <pay by credit card>

ELSE

IF <restaurant accepts cheques> THEN <pay by cheque>

ELSE

<pay by cash>
```

Activity 4 – GOMS

For the example in Activity 3 construct a GOMS model of a customer withdrawing money from a cash machine.

A Discussion on this activity can be found at the end of the chapter.

Keystroke Level Model

The lowest level of GOMS analysis is called the Keystroke Level Model (KLM). This produces quantitative predictions of the time it would take a skilled operator to complete a task. Again, it assumes error-free performance by the operator.

Execution of a task is described in terms of

- 5 physical-motor operators:
  1. **Tk**: (k)eying – how long it takes to press a key (including using modifiers such as the shift key)
  2. **Tp**: (p)ointing – how long it takes to move the mouse (or other such input device) to a target
  3. **Th**: (h)oming – how long it takes to change between input devices e.g. changing between mouse and keyboard
  4. **Td**: (d)rawing – how long it takes to draw a line using an input such as a mouse
  5. **Tb**: click (b)utton – how long it takes to click the mouse button
- **Tm**: (m)ental operator – how long it takes to perform the mental processing for the task
- **Tr**: system (r)esponse operator – how long the system takes to respond

Therefore, execution time for a task is described in terms of the sum of the operators used. For example, suppose we had typed the sentence the quick fox jumps over the lazy dog. Now we want to insert brown just after quick, using a word processor, and assuming that the current point is at the end of the sentence, we need to perform the following steps:

1. move hand to mouse
2. position mouse just after quick
3. move hand to keyboard
4. formulate word to insert - brown
5. type brown
6. reposition insertion point at end of sentence

In terms of the KLM the following operators are needed for the above steps:

1. H (mouse)
2. P, B
3. H (keyboard)
4. M
5. K (b) K (r) K (o) K (w) K (n)
6. H (mouse), M, P, B

So, in total the execution time for this simple task is 3Th + 2Tp + 2Tb + 2Tm + 5Tk (assuming there is no significant response time for the system). Card et al. derived values for the time to complete these operators from empirical studies. These are listed below (for an expert typist), and give a total execution time of 1.2 + 2.2 + 0.4 + 2.7 + 0.6 = 7.1s in this case. As we shall see later in Unit 9, these quantitative predictions of execution time can also be used to compare designs.

Operators	**Time (s)**
Tk	0.12
Tp	1.10
Th	0.40
Td	1.06
Tb	0.20
Tm	1.35

Activity 5 – KLM

Carrying on with the example in Activity 4, imagine that customers could withdraw money using their personal computer. In this case data entry would be via the keyboard, and selection of options would be done using the mouse. Using KLM, work out the execution time for the activity of withdrawing £10 assuming that both keyboard and mouse are to be used, that the PIN is 1234, that it takes the system 10s to return the card and cash, and that £10 is one of the predetermined amounts listed.

A Discussion on this activity can be found at the end of the chapter.

Summary of GOMS

GOMS provides a way of making predictions about the time an expert user would take to complete a task using a given user interface. Furthermore, as GOMS modeling makes user tasks and goals explicit these descriptions could be usefully re-employed in the development of an on-line help system. These descriptions can additionally be used to suggest questions users will ask and the answers in terms of actions needed to complete tasks and meet goals.

However, as mentioned several times before, the tasks must be must be goal-directed, that is the user must have a specific aim in mind. Some activities are more goal-directed than others, but it could be argued that even creative activities contain goal-directed tasks. Furthermore, GOMS assumes that tasks involve routine cognitive skill as opposed to problem solving, and that no errors occur, which is hardly realistic.

Review Question 5

GOMS is a cognitive modelling approach. What does it model, and how?

Answer at the end of the chapter.

Interacting Cognitive Subsystems

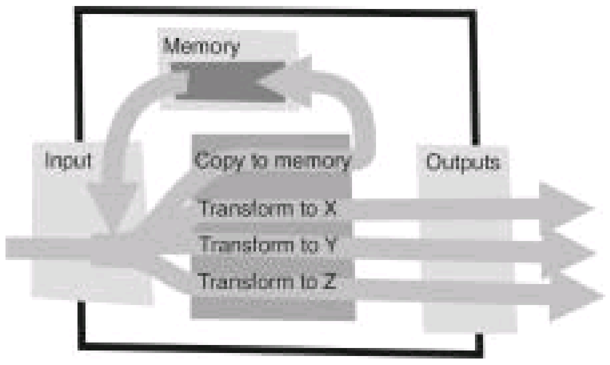
ICS is an elaborate framework which assumes that human perception, cognition, and action can be analysed in terms of discrete, inter-linked, information processing modules. In contrast to GOMS, ICS is a much richer way of modelling human cognition as we shall see in this section.

Subsystems

Each subsystem of ICS is independent and operates in a specific domain of processing of which there are three main components:

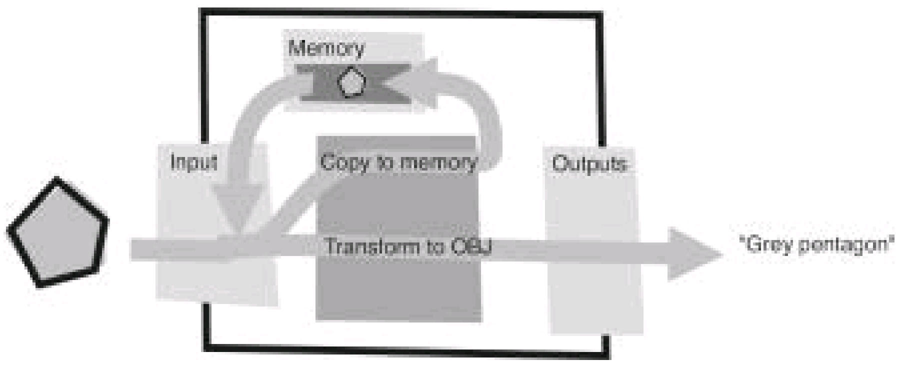
- **sensory** – visual and auditory stimulus
- **representational** – representations of information
- **effector** – body movement

Although each subsystem operates in different domains, and on different kinds of information, they all share a common structure as illustrated in the following diagram. Each subsystem has an input to its left, and one or more outputs to its right, as well as its own memory record and a set of transformations. All information which is input to the subsystem is stored in the memory record (and can at a later point be used as an input to the subsystem). The outputs depend on the transformations which transform information from one form to another. In the example we have three transformation which transform the input information into three different codes – X, Y, and Z.



An ICS Subsystem

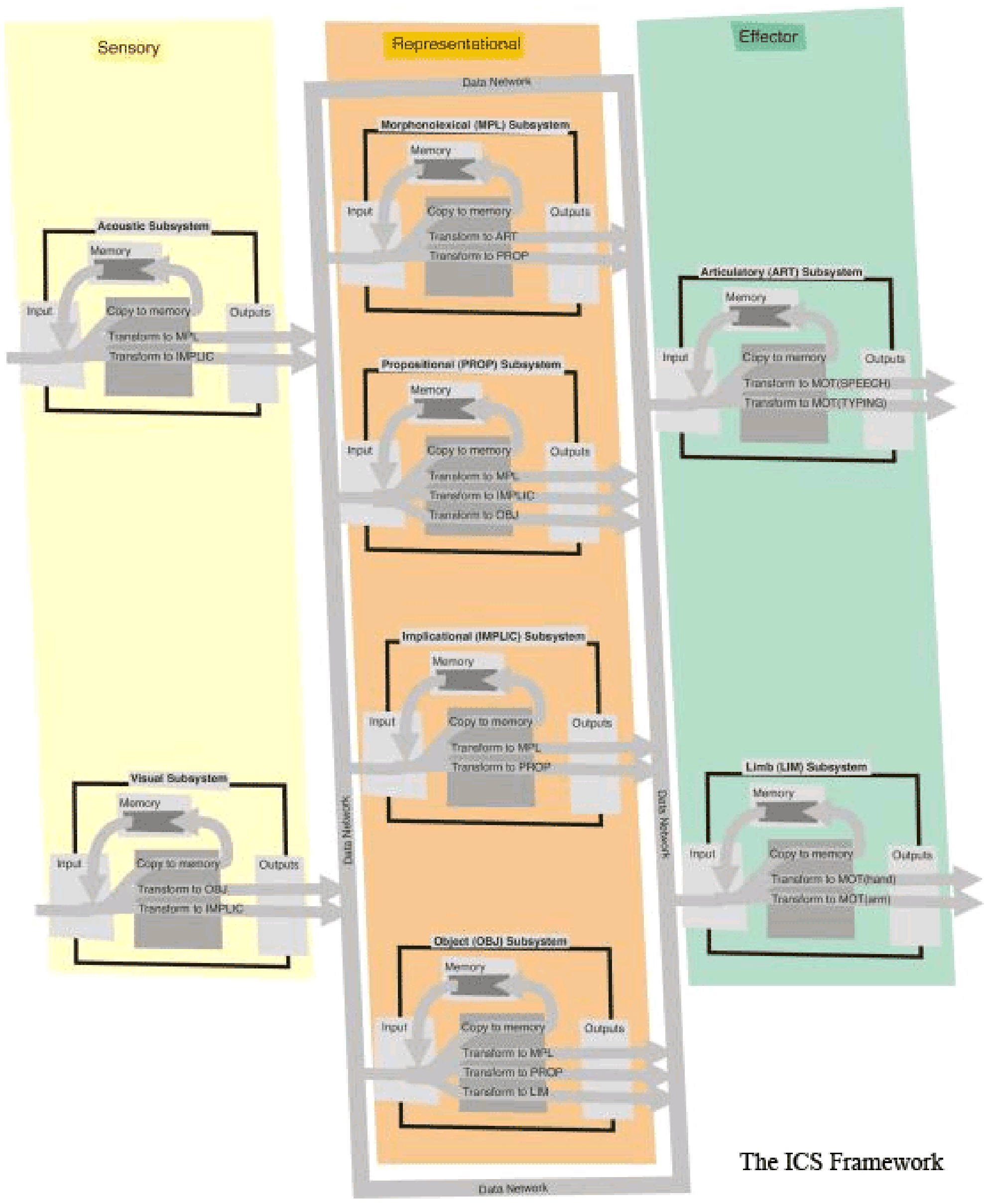
As mentioned previously, ICS models human cognition in terms of several of these subsystems linked together. We shall come to the complete network later, but let’s start to understand the ICS framework by considering the operation of one subsystem. A good point to start understanding ICS is by considering the visual subsystem (illustrated below) which forms part of the sensory domain. This takes input from the eyes in the form of sensory representations with information about edges, angles, shades, contrasts, hues etc. (here the person is looking at the pentagon to the left of the diagram) and transforms it to object representations with information about depth, position, orientation, shapes, etc. (here the object representation describes the image as a grey pentagon). Note that a copy of the visual representation is stored in the memory record for later use. Note also that the clarity of the visual representation alters the success of the transformation to an object representation. It is important to remember that these two representations (visual and object) are both mental representations, but at different levels of information.



The Visual Subsystem

The key to using ICS is the inter-linking of the subsystems. Below is a diagram of the complete ICS framework – the VIsual Subsystem (VIS) just described can be seen in the bottom left of the diagram in the sensory domain i.e. input from the sense. We need to further process information to extract meaning and make decisions e.g. we may need to interpret the grey pentagon as a fifty pence piece and then use it appropriately.

ICS can be used to describe a user’s mental processes whilst performing a task. Such descriptions can then be used to assess the relative amounts of cognitive resources (the numbers of transformations) needed to complete tasks with different interfaces. In the example a user is revising some text in a word processor. Whilst reading the text they notice a particularly difficult sentence and decide to rephrase it. In an ICS description of the cognitive processes involved in achieving this task we might start by considering what the sensory inputs are. As the user has read the text from the screen they are using their visual sensory input. So the first subsystem involved is the Visual Subsystem which transforms the visual representation into an object representation (VIS ® OBJ). This transformation produces information about what letters are being viewed. As the output is in OBJ form it is then processed by the Object Subsystem which performs the transformation OBJ ® MPL. This morphonolexical representation encodes the surface structure of the sentences in a speech based code which discards the surface structure of the words i.e. it encodes what the words are, rather than what letters they are composed of. The MPL representation is then processed by the Morphonolexical Subsystem which performs the transformation MPL ® PROP producing a representation which encodes the meanings of the individual words in the sentence, and the relationships between the words. From this representation the Propositional Subsystem transforms it to an implication representation (PROP ® IMPLIC) in which the meaning of the sentence itself is identified. Several different meanings of the sentence may be identified in which case there may be several iterations of PROP ® IMPLIC (Propositional Subsystem) and IMPLIC ® PROP (Implicational Subsystem) to arrive at a satisfactory understanding of the meaning. At this point the representation needs to be transformed into a form suitable for the Limb Subsystem so that the correction can be made. This would involve the following transformations by Subsystems: IMPLIC® PROP (Implication Subsystem), PROP ® OBJ (Propositional Subsystem), OBJ ® LIM (Object Subsystem), LIM ® MOT(hand) (Limb Subsystem – transforms LIM into actual hand actions).



The ICS Framework

Of course, there is plenty more processing involved in locating the text on the screen, positioning the cursor, reading the menu bar etc. but this description gives and idea of the processes involved. The sequence described can be summarised as follows (Þ indicates external input or output, ® is a transformation, » indicates data transmitted across the data network, and {} enclose a set of transformations which may occur several times):

```
Þ VIS ® OBJ »

» OBJ ® MPL »

» MPL ® PROP »

{» PROP ® IMPLIC »

» IMPLIC ® PROP »}

» PROP ® OBJ »

» OBJ ® LIM »
» LIM ® MOT(hand) Þ
```

Summary of ICS

ICS is intended to provide a detailed description of the cognitive resources needed to use and learn a system. As such it is far more powerful than GOMS, but also much harder to use and to interpret the results it produces. Even trained psychologists would find it hard to consistently determine which Subsystems are involved in users attempting to complete tasks. One approach to alleviating this problem is to develop some sort of software toolset to help evaluators use the framework. This might support activities such as determining which Subsystems are used, and which representations are appropriate.

Review Question 6

ICS provides a detailed model of human cognition which can be used to determine how much cognitive effort a user will have to employ to complete a task. How does this differ to GOMS in terms of its model of human cognition, the intended users of the model, and the kinds of assessments it can make?

Answer at the end of the chapter.

Activity 6 - ICS

ICS is notoriously difficult to use. How would you develop a tool to make its use easier , what functionality would it have, and how would that make using ICS easier?

A Discussion on this activity can be found at the end of the chapter.

Summary of Cognitive Modeling

In this section we looked at models of human cognition which are intended to help designers understand how users would behave with their systems. Both models have a computational feel reflecting their roots in cognitive psychology. Furthermore, both models can be quite difficult to use, especially for large systems

Summary of User Modeling

The two approaches discussed in this unit – user requirements modelling and cognitive modelling – are very different uses of models. The first models the situation in which users work and the constraints their environment places upon them, and therefore new systems, whereas the second tries to give designers some understanding of the working of the human mind. They may be radically different models, but the reason for their existence is the same – to ensure that designs take account of users. This is referred to as user centred design and is an important departure from the conventional approach to design which concentrates on functional requirements of the system. The aim is that by considering the user (either from the point of view of their work context, or their cognitive abilities) the developed systems will be successfully deployed in the workplace and accepted by the users.

Discussion Topics

There are many new concepts in this unit. If you want to discuss with your colleagues or make comments about the concepts to them, use the on-line facilities.

Discussion topics:

- User centred requirements analysis increases the effort of developing systems – what benefits can such approaches bring.

- How would you convince the company you work for to adopt user centred approaches?
- How can some of the more complex approaches (e.g. GOMS and ICS) be made more accessible to evaluators?

Answers and Discussions

Answer to Review Question 1

Models provide us with a representation which is easier to use and manipulate than the thing we are designing (that the model represents). This means that it is easy to test out different designs to see which is best for a given purpose. We can also use models to model a situation e.g. the social situation in which the thing we are designing will be placed. This helps us to understand how the new system will impact on the current working situation. Finally, we can use models to provide approximations of human behaviour which can be used to determine whether our designs will be difficult to use, without involving users.

Models contribute in two main ways – generative or evaluative. Evaluative allow us to assess whether the final product will meet some criteria. Generative models provide input to design – it should be noted that most models have some generative aspect to them.

Answer to Review Question 2

Use centred requirements analysis employs models to represent a typical user’s work environment. This ranges from representations of the stakeholders and their perspectives to considerations of how systems influence the social and organisation situation. Cognitive models, on the other hand, provide a simplistic view of how user might react to situation, and in turn, how they might use a newly designed system.

Answer to Review Question 3

Both socio-technic and SSM model the stakeholders involved in the development of a new system. These models give designers a grasp on the context of who are involved in the development of the new system, and what impact this will have on their working environment. The primary difference between socio-technic and SSM approaches is that SSM takes a wider view of the context. This means that it provides a model of not just the stakeholders, but also the environment in which they work. Providing such a wider view allows designers to get a grasp on the wider picture, but does involve extra effort on the designers’ and analysts’ part to develop and understand the models.

Answer to Review Question 4

The crucial difference between participatory design and socio-technical and SSM approaches is that participatory design is a design philosophy in which users are considered central and are included on the design team throughout the stages of development. Socio-technical and SSM approaches are concerned with establishing models of user requirements (as opposed to necessarily including them in the design process). In contrast, models used in participatory design are used to support communication between designers and users e.g. storyboards which model typical user interactions.

Answer to Review Question 5

GOMS uses a Model Human Processor to model human cognitive processes. This is used in conjunction with models describing Goals, Operators, Methods, and Selection rules to determine execution times using a Keystroke Level Model. This provides predictions of task execution time for tasks that involve no user errors or problem solving.

Answer to Review Question 6

GOMS' model of human cognition is referred to as the Model Human Processor. Both models assume that there are interrelated information processors, but ICS' model is much more complex. Essentially the representational domain of ICS (containing 4 subsystems) is the cognitive processor of GOMS.

GOMS attempts to make quantitative predictions of task execution time by expert users. This takes no account of errors, learning, or problem solving. ICS, on the other hand, can model these issues, but does not give simple quantitative predictions. As such ICS is intended to be used by expert psychologist evaluators, whereas GOMS was intended to be used by trained evaluators.

Discussion on Activity 1

The following is a definition of the stakeholders concerned with the introduction of a new cash machine:

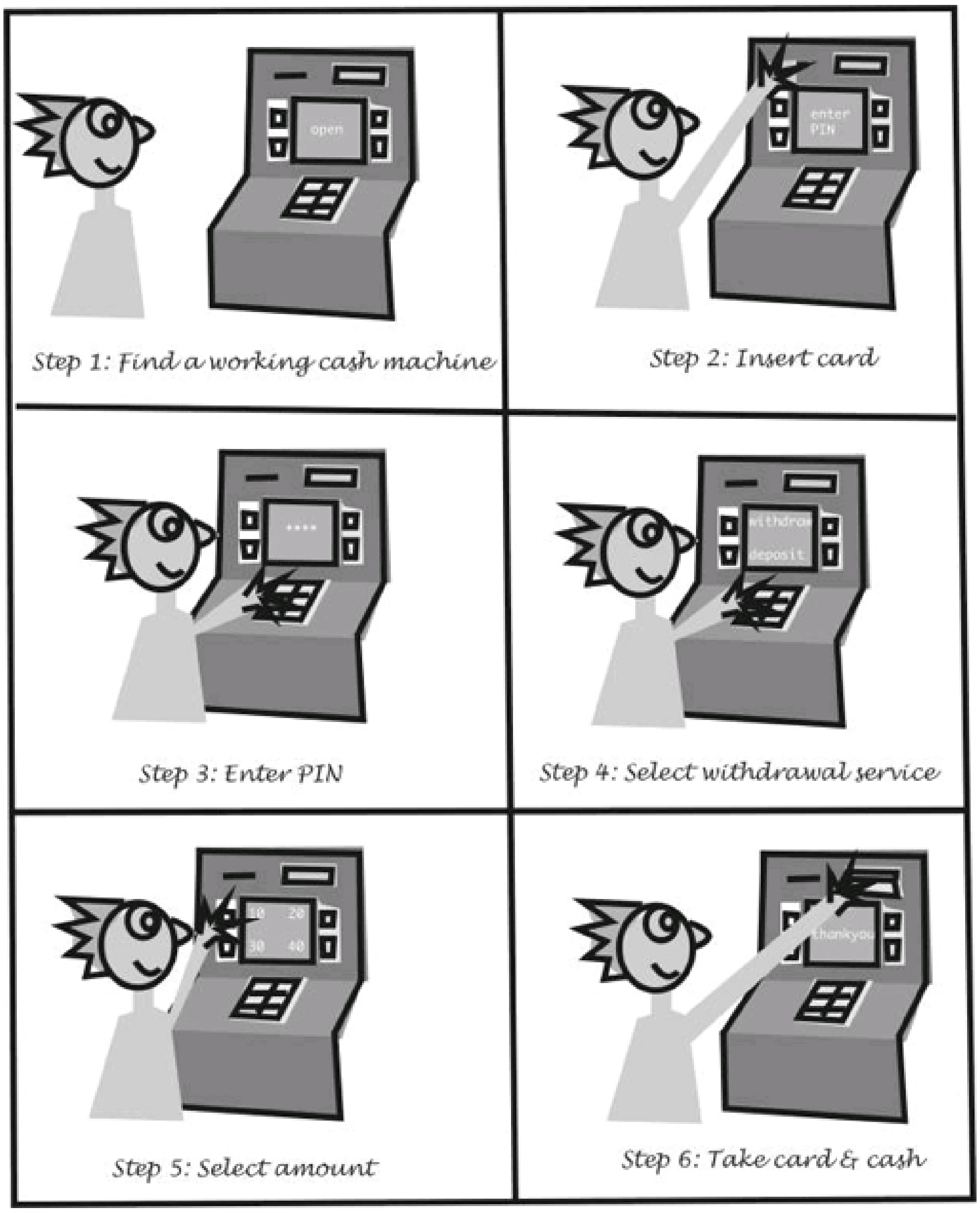
- **Primary** – people with cash cards, employees of the bank who have to service the machine e.g. fill it with money.
- **Secondary** – people such as accountants who receive receipts from the machine, people who receive money from the machine in payment e.g. shop keepers.
- **Tertiary** – managers of the bank and the directors of the bank.
- **Facilitating** – the employees of 'Cash Machines R Us' who are involved with the design, development, and later maintenance of the cash machines.

Discussion on Activity 2

- **Clients** – bank customer, bank clerks, bank manager, bank's board of directors.
- **Actors** – bank customer, bank clerks.
- **Transformations** – for withdrawal of money – transforms a customer's request for cash into cash and possibly a receipt.
- **World view** – for a bank customer – makes it quick and easy to withdraw money at any time of the day.
- **Owner** – owned by the bank, change may be authorised by senior people in the bank such as the board of directors, or maybe the head of information technology for the bank.
- **Environment** – the cash machine must be secure, robust enough to withstand all weather conditions and vandalism, and at a height that is convenient for most customers.

Discussion on Activity 3

A storyboard for this activity is as below:



Discussion on Activity 4

```
Goal – withdraw cash.

  Method - Enter PIN

    Operators – press-1st digit of PIN, press-2nd digit of PIN, press-3rd digit

  Method - Select withdrawal

    Operator – select ‘withdraw cash’

  Method - Select amount

    IF <amount is shown> THEN <Operator - select amount>

    ELSE
```

```
        <Operators - select ‘other amount’, press-1st digit of amount, press-2nd
      IF <3 digit amount of cash> THEN <Operator – press-3rd digit of amount>
    Method - Collect card & money
      Operators – take cash, take card
```

Discussion on Activity 5

GOMS	**Operator**	**Time(s)**
Goal – withdraw R100.		
Method - Enter PIN		
Operators – press-1, press-2, press-3, press-4	4xTk	0.48
Method - Select withdrawal		
Operator – select ‘withdraw cash’	Th + Tp + Tb	1.70
Method - Select amount		
IF <amount is shown> THEN <Operator - select amount>	Tp + Tb	1.30
ELSE		
<Operators - select ‘other amount’, press-1st digit of amount, press-2nd digit of amount>		
IF <3 digit amount of cash> THEN <Operator – press-3rd digit of amount>		
Method - Collect card & money		
Operators – take cash, take card	Th + Tr	10.40
Total(s)		13.88

Discussion on Activity 6

One approach would be to explicitly model each subsystem of ICS and connect them via data pathways. Analysts could then feed in different kinds of information to the sensors, indicated which subsystems they think might be used, and then observe how the data flows around the network. This would reduce the burden of working out how the data flows around the system.

A more comprehensive approach would be to use an expert system (a system which contains a lot of knowledge about an area, and some artificial intelligence) to help work out which subsystems might be involved in processing information. This helps reduce the problems of deciding which subsystems and representations are likely to be used.

Chapter 8. Task Analysis

Table of Contents

Context

In the last unit we looked at user models and modelling. Users usually have goals and carry out tasks to achieve those goals (interacting with a computer system as necessary). In this unit we will look at ways of modelling tasks. There are different types of task model. All of the techniques concern goal-oriented things that people do but some focus on the actions carried out to achieve a goal; some on the knowledge required to carry out a task; and, others on the relationships between the objects involved in the task. This unit will review three major types of model (Hierarchical Task Analysis, Knowledge Based Analysis and Entity Relationship modelling). You will see how they are employed in systems development. We will focus on HTA and you will learn how to produce this sort of analysis and how the results can be used.

Objectives

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

- Explain the purpose of task modeling
- Distinguish between task analysis and other analyses such as GOMs (Unit 7)
- Distinguish between HTA, KB and ER analyses.
- Give a brief review of KB and ER analyses and explain how they can be used.
- Carry out HTA for a range of user goal types.
- Explain and show how results of HTA can be used to improve interaction.

Unit Plan

In previous units you have focused on design (units 2, 3 & 6) and on users (units 4, 5 & 7). But of course users and computers do not exist in isolation: apart from some games applications, computers are there to support users in going about their every-day work. To be useful, computer systems have to help users to do the things they need to do. To be usable, they need to help the users do them in a way that seems natural to those users. Task analysis is concerned with understanding how users go about their work at the moment, so that new computer systems can be designed to help with that work rather than hindering it or forcing it into a different pattern.

In this unit, we will study three different approaches to analysing users' tasks:

- **Hierarchical Task Analysis (HTA)** involves describing users' tasks in terms of the activities involved at different levels of detail. Superficially, it looks a bit like the GOMS analysis that you studied in unit 7 but, as you will see later in this unit, it is concerned with larger-scale tasks in the work domain, rather than user tasks (involving the details of thinking and acting with a particular device).
- **Knowledge-Based (KB) Analysis** involves creating taxonomies of objects that are important within the work domain.
- **Entity-Relationship (ER) Modelling** involves describing the entities (objects and actors) involved in the domain and relationships between those entities. You may have come across this type of approach before, if you have studied systems analysis, but don't worry if you haven't.

In this unit, we will cover how to create each of these types of representation of a task and how to use that representation to reason about design. We will focus most on HTA.

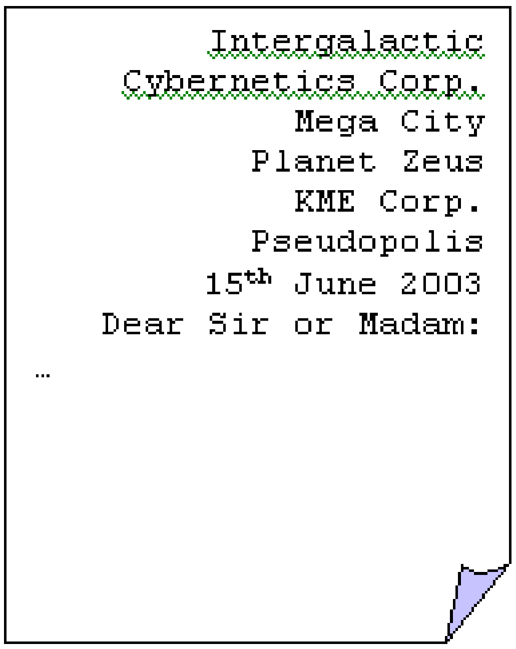
Introduction to this Topic

W'll start with a simple example, based on a simple task of writing a letter and preparing it for posting.

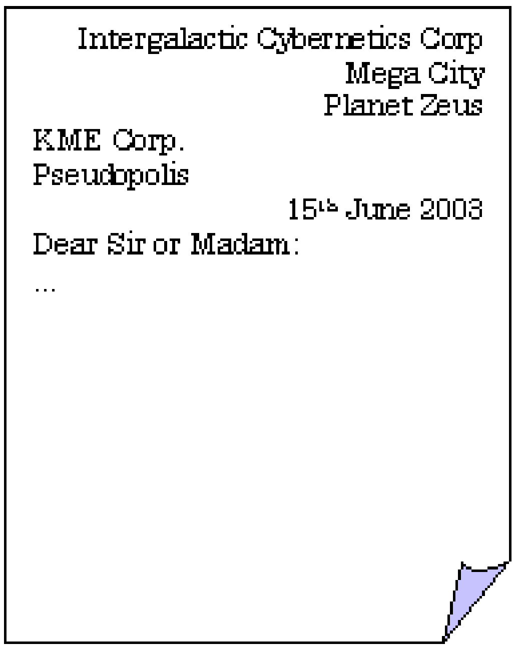
First, we observe Linden, who has no computer support, performing this task in her office. She finds a sheet of paper, a matching envelope and a pen, and settles down at her desk to write. She starts by putting her own address in the top right-hand corner, then picks up her address book (which is lying on the desk beside her) and flicks to the page containing the addressee's address information. She copies the addressee's address to the paper (left-hand side), then writes today's date and starts on the body of the text:Dear...". When she has finished, she signs the letter then reaches for the envelope, writes the name on it and copies the address from the address book onto the envelope, folds the letter, puts it into the envelope and seals the envelope. Task completed.

We then go to the next office to observe Remi using his word processor to do something similar. He settles down in front of his computer and opens his word processing package, using a document template that already includes his own address and today' s date. He opens his electronic address book, finds the appropriate address, and uses his mouse to copy and paste the address from the address book into his document. He starts on the body of the text:Dear...". When he has finished composing the letter on the screen, he selects the print option and waits until a message appears on his screen to say Your document Letter to KME has been printed on Laser 6.He gets up and walks out to the shared printer area, heads for the printer in the far corner, and flicks through the pages that have accumulated there until he finds his letter. He picks it up, walks back to his desk, reads it through and signs it. The printer does not print on envelopes, so he finds a sheet of mailing labels that will fit through the printer. He has a document template that is set up to print in the format appropriate for mailing labels, so he opens that and copies the information from his electronic address book to this new file. He selects the print option, remembering to select the option for manual feed (so that he can place the blank mailing label sheet in the manual feed tray of the printer). He then walks out to the printer and inserts the mailing labels in the manual feed tray then hurries back to his office to click on the OK button that has now appeared next to the message manual feed: click OK when ready to print. He walks back to the printer again to collect the printed label, then goes to find an envelope. He folds the letter and slips it into the envelope, sticks the mailing label on the envelope, then seals it. Task completed.

Which of these procedures is better? It depends on what the aims of the letter-writing exercises are. The word-processed version will look more professional, whereas the hand-written one will look more personal. The word-processed letter may be easier to read (depending on how legible Linden's handwriting is). Which was produced more efficiently? That probably depends on how quickly Linden writes, and how quickly Remi types. For short letters, it is quite likely that the hand-written one will have been produced more quickly. Computerisation does not always improve efficiency.



Intergalactic
Cybernetics Corp.
Mega City
Planet Zeus
KME Corp.
Pseudopolis
15th June 2003
Dear Sir or Madam:
...



Intergalactic Cybernetics Corp
Mega City
Planet Zeus
KME Corp.
Pseudopolis
15th June 2003
Dear Sir or Madam:
...

Is task analysis about reproducing exactly the same situation with a new design as the one we are replacing? Absolutely not! Intelligent design is about adapting and improving the work system. So we see in this small example that early on there were things that could be done more efficiently with the word processor than by hand: by integrating address book information with the letter-writing tool, the act of transcribing could be made more efficient. You can also imagine that if this letter were a fairly standard one, Remi could have cut and pasted text from another document to speed up the writing of this one. However, later stages (which involved integrating tasks with the word processor with printing tasks) were, frankly, tedious: although we feel that it ought to be easier to print a mailing label from pre-existing information, in practice the system used by Remi does not support this.

Domain tasks and device tasks

In the discussion of letter-writing, we saw that very similar end objectives could be achieved in very different ways using different tools (pen or word processor plus printer). The two scenarios described similar domain tasks – what the user is achieving in the world – but involved different device tasks – what the user had to do using the available tools to achieve that effect. One of the most important challenges of design is often to ensure that there is a good ‘fit’ between the domain tasks and the device tasks. In an ideal world, the device would be almost invisible to the user – not in the sense that it cannot be seen, but in the sense that it is not noticed.

We can draw on an analogy of a hammer: if you are hammering in a nail, things always go best when you are unaware of the hammer itself; it is just an extension of your hand and the hammering action is fluid and easy. Then you bang your finger, and suddenly that flow of activity is broken; you are intensely aware of the hammer as a separate object, and you have to work hard to restore the earlier productive hammering action. Tools should be ready-to-hand in the way that a hammer is to a carpenter: the user should not be fighting with the tool, but using the tool easily and naturally to achieve their tasks, almost unaware of its existence. For Linden, the worst things that might happen are her pen leaking or running out of ink. Remi’s work is more prone to the tools getting in the way of the task, although, conversely, his tools may also provide him with more active support for those same tasks.

Of course, there are cases where the computer system is the focus of the activity: a good computer game is engrossing, but it does depend on the computer system being there and visible: it is the focus of attention. However, even in this case, the user should be able to focus on the game rather than being distracted by pictures going jittery, the sound fading unpredictably or other unintended features.

Of course, games are not the usual focus of task analysis, which is more commonly concerned with the productive kinds of tasks that are found in work environments (offices, factories, etc.). In the remainder of this unit, we will focus our attention on the kinds of work tasks that have a clear goal – typically an end product.

Data Acquisition

Task analysis of any kind starts from data – information about the work and the work setting. For HTA the data that is needed is data about procedures – about how activities are structured into tasks. For ER and KB analysis, the data that is needed concerns objects in the work domain and their interrelationships, and maybe also the actors (people that are essential to getting the work done).

In the scenarios above, we talked about going into Linden’s and Remi’s offices and observing them. Implicitly, while observing them we were also taking notes about their activities that could be used later for analysis. Observation and note-taking (or even video-recording) are one way of gathering the data that is needed for task analysis. Another way of getting such information is by interviewing users (i.e. the people who perform the tasks being studied). Again, responses can be noted by hand, or recorded using a tape-recorder. Each of these techniques has advantages and disadvantages.

Interviewing is often more efficient than observation. As people talk about their work, they use a particular vocabulary that expresses their understanding of what they do and how they work that can usefully be adopted in any computer support for their tasks. You find out how they think they perform their jobs and the things that they perceive as being important.

Unfortunately, people are often unable to articulate what they really do. This may be because they are so immersed in their working environments, so familiar with the way things work, that they take much of their knowledge for granted. Often the most important things about the way they work are the ones that are most obvious to them, and therefore the ones they will forget to mention. Also, as they become experts at their tasks, they forget all the details: they develop ‘compiled skills’ which becomes difficult for them to break down and describe fully. To take a simple example that may be familiar to you: if you are driving a manual car, as you pull away from a junction you will change up the gears. When do you change gear (e.g. from third to fourth)? What is the detailed procedure you follow as you change? Do you take your eyes off the road for any reason? Learner drivers are very aware of the procedure; each foot movement, each glance in the mirror, the act of feeling for the gear lever (maybe looking for it) are all deliberate actions. As expertise develops, the conscious awareness of the process diminishes. To understand users’ tasks in detail, it is often necessary to observe them, to identify taken for granted knowledge and compiled skills.

A third source of data for task analysis is often existing documentation: there may be procedure documents that describe how tasks should be performed (which may or may not describe how they are actually performed). In what follows, we are assuming that data has be acquired from somewhere, and we focus on ways of analysing that data.

Design Evolution and Revolution

Before we start, though, a note of caution about the limits of task analysis: it’s no good for revolutionary design. Every now and then, a totally new kind of design appears on the scene – one that really changes the way we do our work, or think about interaction. Two of the examples from recent years that you will be familiar with are the Graphical User Interface (GUI), which moved us forward from the text-based command-line interfaces of an earlier generation and made computing much more accessible to non-specialists, and the World Wide Web, which has revolutionised information access and hastened the globalisation of interactions and commercial transactions. For every high-impact success like these, there are untold numbers of failures – revolutionary ideas that simply failed to catch on and faded into oblivion.

Contrast these with the word processor, which evolved from the typewriter and retained many features of its predecessor: each generation of word processor has moved a little further from its origins, as users find new uses for the current generation of machine and adapt their behaviour to the new possibilities, which are then ‘designed in’ to the next generation products, so that we get co-adaptive behaviour (with users adapting to new systems, which are then re-designed to support new uses, which…)

Task analysis provides good support for design evolution, where understanding of the current task – the task structure, the entities and actors – is used as a starting point for new design. It is no use for design revolution, which typically involves creating new possibilities, that simply did not exist before, and introducing new concepts (e.g. the graphical object or the hypertext link).

Hierarchical Task Analysis (HTA)

Hierarchical Task Analysis has been in use for a long time – since the 1960’s or even earlier. It is most suitable for analysing tasks that have a well-defined structure – that is, tasks which tend to be performed in similar ways every time, rather than those that have a very loose structure. HTA involves describing the task in terms of a task-subtask hierarchy and a set of plans that define in what order subtasks may be performed, or under what circumstances particular subtasks are performed at all.

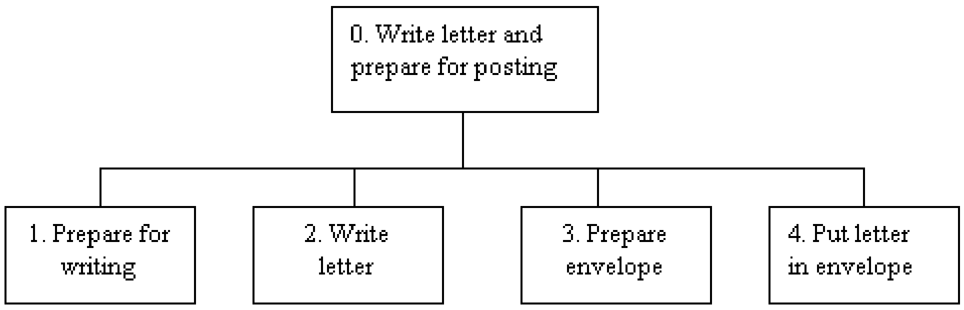
Example

We will start with the simple example introduced earlier: writing a letter and preparing it for posting. Initially we consider the description of Linden's activity:

Description of task	**Subtasks**
First, we observe Linden, who has no computer support, performing this task in her office. She	Prepare for writing.

Description of task	**Subtasks**
finds a sheet of paper, a matching envelope and a pen, and settles down at her desk to write.	
She starts by putting her own address in the top right-hand corner, then picks up her address book (which is lying on the desk beside her) and flicks to the page containing the addressee's address information. She copies the addressee's address to the paper (left-hand side), then writes today's date and starts on the body of the text: "Dear …". When she has finished, she signs the letter	Write letter
& then reaches for the envelope, writes the name on it and copies the address from the address book onto the envelope, &	Prepare envelope
& folds the letter, puts it into the envelope and seals the envelope. Task completed.	Put letter in envelope

One of the standard ways of presenting a HTA is as a tree structure:


0. Write letter and prepare for posting
1. Prepare for writing
2. Write letter
3. Prepare envelope
4. Put letter in envelope

There may be some other notations you are familiar with where the order of appearance of boxes in the tree indicates ordering (probably left to right). This is not the case for HTA: the only thing that matters is the hierarchy. As well as presenting the hierarchy, it is necessary to describe plans that define the possible ordering of activities. In this case, a suitable plan would be:

Plan 0: Do 1, then 2 and 3 in either order, then 4.

Although tree structures are visually appealing – well, more appealing than the alternatives, anyway – they can be tedious to draw without a suitable tool. Therefore an alternative text-based notation that relies on indentation is often used. We will use this textual notation to expand the task description for the letter-writing task.

- 0: Write letter and prepare for posting
  - 1: Prepare for writing
    - 1.1: Get paper
    - 1.2: Get envelope
    - 1.3: Get pen
    - 1.4: Get address book (not explicitly stated, but clearly necessary)
  - 2: Write letter
    - 2.1: Write own address

    - 2.2: Write addressee's address
    - 2.3: Write date and "Dear..."
    - 2.4: Write body text of letter
    - 2.5: Sign off
  - 3: Prepare envelope
    - 3.1: Write name on envelope
    - 3.2: Write address on envelope
  - 4: Put letter in envelope
    - 4.1: Fold letter
    - 4.2: Place letter into envelope
    - 4.3: Seal envelope

Again, we need plans to describe how to perform each subtask:

- Plan 1: Do 1.1, 1.2, 1.3 and 1.4 in any order
- Plan 2: Do 2.1 then 2.2 then 2.3 then 2.4 then 2.5
- Plan 3: Do 3.1 then 3.2
- Plan 4: Do 4.1 then 4.2 then 4.3.

Task analysis involves generating as general a description as possible. So, for example, we might want to generalise tasks 2.1, 2.2 and 2.3 to a new task: write head of letter. Similarly, we might notice that it is not necessary to have the envelope to hand until the time when it is to be prepare, or the paper to hand until the point where the user starts writing the letter, but we need the pen and address book for both, so we might break down task 1. If we do these things, we get a new structure:

- 0: Write letter and prepare for posting
  - 1: Get paper
  - 2: Get envelope
  - 3: Prepare for writing
    - 3.1: Get pen
    - 3.2: Get address book
  - 4: Write letter
    - 4.1: Write head of letter
      - 4.1.1: Write own address
      - 4.1.2: Write addressee’s address
      - 4.1.3: Write date and “Dear…”
    - 4.2: Write body text of letter

    - 4.3: Sign off

  - 5: Prepare envelope
  - 6: Put letter in envelope

Again, we need plans to describe how to perform each subtask:

- Plan 0: Do 1, 2, 3, 4 and 5, then 6. 3 must be done before 4 and 5; 1 must be done before 4; 2 must be done before 5.
- Plan 3: Do 3.1 and 3.2 in either order
- Plan 4: Do 4.1 then 4.2 then 4.3.
- Plan 4.1: Do 4.1.1 then 4.1.2 then 4.1.3
- Plan 5: Do 5.1 then 5.2
- Plan 6: Do 6.1 then 6.2 then 6.3.

We see that now different parts of the task analysis are presented at different levels of detail. This is often thought of as a Bad Thing, but in this case it allows us to describe the optionally and alternative orderings within the task more clearly. As with most aspects of design, there is no perfect solution just solutions that are better or worse for particular purposes.

Exercise 1 HTA for word processing letter

Look at the paragraph that describes Remi's word-processing task (reproduced below). Mark up this paragraph into activities that sensibly constitute a subtask, using the analysis already presented for Linden's activity as a guide. Construct a HTA description of this task.

Note

We then go to the next office to observe Remi using his word processor to do something similar. He settles down in front of his computer and opens his word processing package, using a document template that already includes his own address and today's date. He opens his electronic address book, finds the appropriate address, and uses his mouse to copy and paste the address from the address book into his document. He starts on the body of the text: "Dear …". When he has finished composing the letter on the screen, he selects the 'print' option and waits until a message appears on his screen to say "Your document Letter to KME has been printed on Laser 6". He gets up and walks out to the shared printer area, heads for the printer in the far corner, and flicks through the pages that have accumulated there until he finds his letter. He picks it up, walks back to his desk, reads it through and signs it. The printer does not print on envelopes, so he finds a sheet of mailing labels that will fit through the printer. He has a document template that is set up to print in the format appropriate for mailing labels, so he opens that and copies the information from his electronic address book to this new file. He selects the print option, remembering to select the option for 'manual feed' (so that he can place the blank mailing label sheet in the manual feed tray of the printer). He then walks out to the printer and inserts the mailing labels in the manual feed tray then hurries back to his office to click on the "OK" button that has now appeared next to the message "Manual feed: click OK when ready to print". He walks back to the printer again to collect the printed label, then goes to find an envelope. He folds the letter and slips it into the envelope, sticks the mailing label on the envelope, then seals it. Task completed.

Using the guidelines for adaptation, and the example of modifications presented above, modify your HTA so that it better matches the general task of using a word processor such as that being used by Remi

Answer at the end of the chapter.

Exercise 2 Re-designing computer-based procedure

You should now have HTAs for two similar tasks: writing a letter by hand and word-processing a letter. Hopefully, they share many of the high-level tasks in common, and differ mostly in the detail. Use these HTAs as a starting point for adapting the computer-based task so that it is easier for users.

Answer at the end of the chapter.

Activity 1 - HTA for finding information

Construct a HTA for finding information on the Word Wide Web, starting with data gathering. If possible, you should observe a few other people (fellow students or friends) performing the task, and note down everything they do. If this is not possible, perform the tasks yourself, and note down everything that you do. Use this data as a basis for constructing a HTA. The information seeking should cover two different kinds of information seeking tasks, as described here.

- There is a shareware text on Task Analysis, written by Clayton Lewis and John Rieman, available on the web. Use the Google search engine (http://www.google.com [http://www.google.com/]) to start the search, and observe how people behave from there. When the text has been found, you might like to read some of it!
- Imagine you are planning a trip to Sydney (Australia) or Boston (USA). You need to identify a cheap flight and also find a suitable hotel to stay in. Get all the information that you would need to be able to just book over the Web or telephone to make a reservation.

Use the HTA produced as a starting point for thinking about how the searching task can be made as easy as possible for users. Note that the design of search engines is a separate topic that is beyond the scope of this course!

A Discussion on this activity can be found at the end of the chapter.

HTA Conclusion

One of the hardest things to grasp about HTA is how to construct a ‘good’ one. In one respect, HTA is very easy: do some hierarchies and some plans. In another, that superficial simplicity hides a deeper challenge: what makes a ‘good’ HTA is its fitness for purpose, and assessing that is usually difficult. For this reason, it is important to construct HTA descriptions, and then to modify and rework them, thinking about alternative structures, and about what level of detail it is most useful to go to.

As noted above, HTA differs from GOMS in that it considers only physical activity (not cognitive tasks), and focuses on plans (rather than selection rules). Conversely, large-scale tasks can be described using HTA whereas only a dedicated person with too much times on their hands would complete a GOMS analysis for such large tasks (it would take weeks of effort to do it thoroughly). If using HTA as a guide for implementation or for detailed documentation, a fairly fine grain of detail (down to the individual action level) may be necessary; where it is being used to help get an understanding of the domain tasks, less detail may be appropriate.

Review Questions

Review Question 1

A HTA description consists of two main components. What are they?

Answer at the end of the chapter.

Review Question 2

HTA is a type of Cognitive Task Analysis. True or false?

Answer at the end of the chapter.

Review Question 3

What is the starting point for HTA?

Answer at the end of the chapter.

Review Question 4

Describe the process of constructing a HTA description.

Answer at the end of the chapter.

Review Question 5

Conduct a HTA for an imaginary fast-food delivery outlet (e.g. pizza delivery), from the point where the customer rings in to place an order to the point where the person who has made the delivery returns with the payment.

Answer at the end of the chapter.

Knowledge Based (KB) Analysis

Whereas HTA is concerned almost entirely with procedures, KB analysis focuses on the things (at least in a loose sense) in the task domain, namely objects and actions. KB analysis involves creating taxonomies, or hierarchies, of objects or actions. Such taxonomies can be used in interface design – for example, by ensuring that related objects or actions are grouped together sensibly, and that all important objects are represented at the interface.

Example

Again, we will start with the simple example introduced earlier: writing a letter and preparing it for posting. Initially we consider the description of Linden's activity:

Description of task	**Objects**	**Actions**

First, we observe Linden, who has no computer support, performing this task in her office. She finds a sheet of paper, a matching envelope and a pen, and settles down at her desk to write.	`Paper`
	
	`Envelope`
	
	`Pen`
		`Find`

She starts by putting her own address in the top right-hand corner, then picks up her address book (which is lying on the desk beside her) and flicks to the page containing the addressee’s address information. She copies	`Address (sender)`
	
	`Address (addressee)`
	
	`Address book`
	
	`Page`
		`Pick up`
		
		`Open & flick`

Description of task	**Objects**	**Actions**

the addressee's address to the paper (left-hand side), then writes today's date and starts on the body of the text: "Dear …". When she has finished, she signs the letter	`Date`
	
	`Body text`
	
	`Signature`
		
		`Write`
		
		`Sign`

& then reaches for the envelope, writes the name on it and copies the address from the address book onto the envelope, &	`Envelope`
	
	`Address (addressee)`
		`Pick up`
		
		`Write`

& folds the letter, puts it into the envelope and seals the envelope. Task completed.	`Letter`
	
	`Envelope`
		`Fold`
		
		`Insert`
		
		`Seal`

Looking at this list, we can immediately see that there are two kinds of objects: the physical objects with which Linden is working, and the component parts of the letter (the bits of writing that turn a blank sheet of paper into a letter). Arguably, there are other distinctions – for example, between the address that is a kind of thing-in-the-world (as used, for example, by the postal service) and the address as a small portion of text that contains information about the corresponding thing-in-the-world. Conversely, maybe the actions 'find' and 'pick up' are essentially the same for this task: both are concerned with the user being in possession of the associated object.

A first attempt at a taxonomy might look like this. We are using the terms AND, OR and XOR to indicate whether an object can be or have all the sub-components at the same time. Here, AND means 'all of', OR means 'any number of', while XOR means 'exclusive or, exactly one'.

```
Letter-writing object XOR

   Physical object XOR

      Paper XOR

         Blank sheet
         Letter

      Pen
      Envelope
      Address book

         Page (lots!)

   Portion of text XOR

      Address AND

         Owner XOR

            Sender
            Addressee

         Location XOR

            Letter
            Envelope
      Date
      Body text
      Signature
```

In this particular case, there are no ‘OR’ objects: most parts of the taxonomy are XOR (e.g. the address owner is either the sender or the addressee, and we are not considering the possibility that these might be the same), while we are also saying that an address has both an owner AND a location (where it is written). There are some quirks – for example, that a sheet of paper is either blank or a letter; we might wish to add a third option (partially written). We will not develop this taxonomy further, but quickly present the corresponding taxonomy for actions:

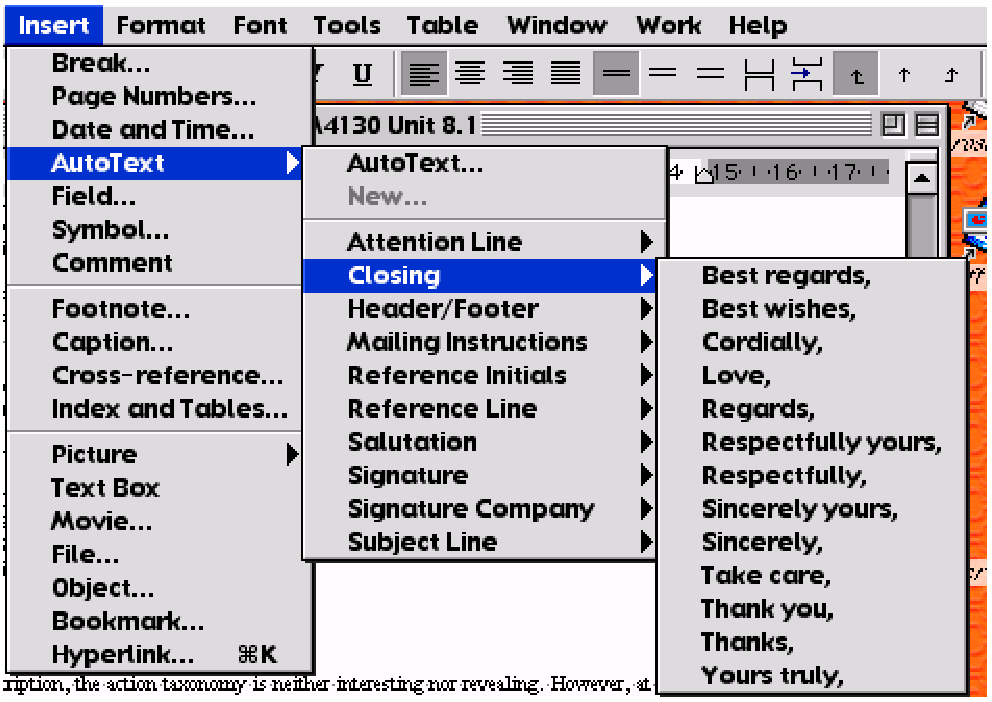
```
Letter-writing action XOR

   Gain possessionof
   Find (e.g. locate address in address book)
   Write XOR

      Write header information
      Write general text
      Sign

   Fold
   Insert
   Seal
```

At this level of description, the action taxonomy is neither interesting nor revealing. However, at a finer grain of detail it could be very valuable. Consider, for example, the pull-down menus implemented within word processors such as MS Word. If you look at the organisation of the menus under the top-level headings (‘file’, ‘edit’, ‘insert’, etc.), the groupings of items within each menu, and the options made available, you will see that there is a strong taxonomic basis to the organisation. See, for example, the Figure below. You can go one further and identify places where the taxonomic structure might be open to challenge (e.g. where should ‘insert table’ be located? And why does this ambiguity arise?).



As discussed for HTA, the level of detail required depends on the purpose. In this figure, we see a fine grain of detail (listing out all the most common closing phrases for letters), which is useful for this level of implementation. In other cases, the challenge may be just to create a taxonomy of the physical objects in the domain to support reasoning about design alternatives, or to support the generation of overview documentation.

Exercise 3 KB analysis for the word-processor version of the letter writing task

Compare the KB description of Linden's task of handwriting a letter to the informal description (presented earlier) of Remi's task of word-processing one. Modify the object taxonomy presented for handwriting to describe the word-processing version of the task. Are there important design points that come out of this?

Answer at the end of the chapter.

Activity 2 taxonomic assessment of e-commerce sites

Select three e-commerce sites (e.g. a bookseller, a travel agent, a financial services supplier) and probe them to see whether you can identify any underlying knowledge based structure. What are the concepts the user has to work with? What actions are available to them? Are actions and objects grouped appropriately at the interface, or within the overall site?

A Discussion on this activity can be found at the end of the chapter.

Review Question 6

What are the core concepts expressed in a KB analysis?

Answer at the end of the chapter.

Review Question 7

KB analysis can be used to guide the design of manuals to support activity. True or false?

Answer at the end of the chapter.

Entity Relationship (ER) Analysis

Entity-relationship analysis is normally associated with systems analysis (notably database design) and, more recently, object-oriented programming, so if you have studied these topics, much of this section might seem familiar to you. Be warned, however: the task analysis we are considering here should have a much broader scope than ‘traditional’ ER modeling, being concerned with the entire work domain and not just the entities that will eventually be represented within the computer system.

Like KB analysis, ER analysis starts with the objects and actions involved in the domain. However, as well as the objects, we are concerned with their properties (and the distinction between objects and their properties, or attributes, is clearer in ER analysis than in KB analysis). Also, we are not concerned with the similarity (or otherwise) between objects, but with how they are related – for example, which actor in the domain performs particular actions on the objects.

Example

Yet again, we will start with the simple example introduced earlier: writing a letter and preparing it for posting. Since Linden's activity involves only one actor, we will only consider the more interesting case of Remi's activity using a word processor. We do something which might at first sight appear strange, namely to include some of the machines as actors, when they behave in a way that is partially autonomous:

Description of task	**Objects**	**Actors**

… Remi settles down in front of his computer and opens his word processing package, using a document template that already includes his own address and today's date. He opens his electronic address book, finds the appropriate address, and uses his mouse to copy and paste the address from the address book into his document. He starts on the body of the text: "Dear …". When he has finished composing the letter on the screen, he selects the 'print' option and waits until a message appears on his screen to say "Your document Letter to KME has been printed on Laser 6".	`Word processor`
	
	`Template`
	
	`Address`
	
	`Date`
	
	`Document`
	
	`Address book`
	
	`Body text`
		`Remi.`
		
		`Printer`
		
		`Computer`

He gets up and walks out to the shared printer area, heads for the printer in the far corner, and flicks through the pages that have accumulated there until he finds his letter. He picks it up, walks back to his desk, reads it through and signs it.	`Printer`
	
	`Page`
	
	`Letter`
		`Remi`

The printer does not print on envelopes, so he finds a sheet of mailing labels that will fit through the printer. He has a document template that is set up to print in the format appropriate for mailing labels, so he opens that and copies the information from his electronic address book to this new file. He selects the print option, remembering to select the option for 'manual feed' (so that he can place the blank mailing label sheet in the manual feed tray of the printer). He then walks out to the printer and inserts the mailing labels in the manual feed tray then hurries back to his office to click on the "OK" button that has now appeared next to the message "Manual feed: click OK when ready to print".	`Envelope`
	
	`Label`
	
	`Template`
	
	`Address book`
	
	`Manual feed tray`
	
	`OK button`
	
	`Message`
		`Printer`
		
		`Remi`
		
		`Computer`

He walks back to the printer again to collect the printed label,	`Printer`	`Remi`

Description of task	Objects	Actors

then goes to find an envelope. He folds the letter and slips it into the envelope, sticks the mailing label on the envelope, then seals it. Task completed.	`Remi`
	
	`Computer`	

The list of objects and actors presented so far may well be incomplete. In one respect this does not matter: task analysis is concerned with helping to get a good understanding of the task domain, and not necessarily with laying things out consistently and completely enough to support implementation. In another respect this is just a starting point: as the analysis proceeds, it becomes gradually more obvious what things matter and what don’t, and we may want to expand detail in some places, while ignoring other issues.

There are already some obvious relationships – for example, that a printed page belongs to someone, that the computer displays a message to a person, that an envelope has a label (maybe), etc.. The process of KB task analysis involves identifying many more such relationships.

If we move on to consider actions, we note that actions are performed by an actor, who may use some instrument (or tool), and they change the state of some object. So, towards the end of the first stage of activity (as presented here), the printer is an actor that prints (action) the letter (object), whereas later the printer is an object into which Remi (actor) places the manual feed tray (object) containing mailing labels.

Exercise 4 ER analysis for the word-processor version of the letter writing task

Develop the ER description of the word-processing task using the ideas as presented in section7.5 of Dix et al. Are there important design points that come out of this? How does this description compare to the KB analysis you conducted earlier?

Answer at the end of the chapter.

Activity 3 entity - relationship assessment of e-commerce sites

Use the same three e-commerce sites as you used for Activity 2. This time, probe them to identify the underlying entities and relationships. Does this raise any new design issues that did not emerge during activity 2?

A Discussion on this activity can be found at the end of the chapter.

Review Question 8

What are the different kinds of entities that are generally listed in an ER analysis?

Answer at the end of the chapter.

Review Question 9

What is the most important difference between ER analysis as a form of task analysis and ER analysis for database design?

Answer at the end of the chapter.

Review Question 10

List some of the important relationships in ER analysis.

Answer at the end of the chapter.

Uses of Task Analysis

We have already summarised most of the important uses of task analysis:

- A source for generating documentation. By structuring the understanding of the task, it becomes much easier to structure a presentation of the task, in user-oriented documentation, whether that be structured around procedures, actions or concepts.
- A source for designing tutorial material. Like documentation, good, user-centred tutorial material that helps users learn to use a product is based around their tasks, so task analysis is a good starting point for designing effective tutorial material.
- Guiding system design. By focusing attention on the current system (and its strengths and weaknesses), task analysis can be used to design new interactions that have evolved in a reasonably natural way from existing practices, and to identify domain objects that need to be represented at the interface, and ways of grouping those objects.
- Requirements capture. Although task analysis refers to the existing system, rather than the planned one, it can help to structure requirements acquisition, particularly as users will often refer to the existing (familiar) system when discussing future requirements. In particular users may not find it easy to list features that should remain unchanged from the existing system, so the task analysis can help focus on what should stay the same as well as what should change.

To summarise: task analysis is necessary for bringing domain knowledge into the design, to make a new design or procedure as familiar and sensible and hence learnable as possible. Task analysis is not easy but then, neither is good design! It takes practice, and it also important to understand both the uses and the limitations of the various task analysis techniques available.

Discussion Topics

Use of HTA

How, if at all, could HTA be useful in designing e-commerce sites?

Comparing KB and ER analysis

Which, to you, seems the more useful technique: KB or ER analysis?

Answers and Discussions

Answer to Exercise 1

You should have a HTA description that describes hierarchies and plans for this task. You can compare this HTA with the handwriting one to identify the important ways in which the task has been changed when it was computerised. Which tasks have been changes substantially for example, made much longer or shorter, or with a different order of subtasks?

One of the purposes for HTA is to help design task structures that seem as natural as possible to users. One aspect of this might be to retain as much as possible that is already familiar to users; another might be to avoid unnecessary additional tasks.

Answer to Exercise 2

Adaptations are likely to include structural changes: for example, should each user have their own printer? Should printers have a special facility for printing labels, or even envelopes, without manual intervention?

Answer to Exercise 3

The word-processor taxonomy is likely to include labels as well as envelopes, owners of printed sheets as well as letters, and printers instead of pens. The multiplicity of printers with different names might be a design issue, as might the kinds of things that can be printed (printable items?). There are more design issues to be discovered!

Answer to Exercise

You should have listed some actors (a particular type of object) and actions they perform. You should have some events (such as 'when the letter has been printed'). You should have some action-event relations (e.g. the label must be put in the manual feed tray before it can be printed). For all the types of relationships, see if you can find at least one relationship in this word processing scenario.

By now, you have probably worked on this letter-writing task as much as you can face. Don't spend too long on this; what matters is that you are starting to get a feel for what is involved in conducting an ER analysis, and how it can help you understand more about the task domain.

Answer to Review Question 1

Hierarchical task structures and plans.

Answer to Review Question 2

This statement is false. Cognitive Task Analysis techniques focus on the details of what is going on in the head and small-scale actions. GOMS is an example of a CTA technique. It has the same type of hierarchical structure as HTA, but has selection rules instead of plans.

Answer to Review Question 3

HTA, like any task analysis, start from data about how people perform tasks. This data may be gathered by observation, interview, or by reading documentation.

Answer to Review Question 4

Construction involves structuring and re-structuring the description (hierarchy and plans) to make it as meaningful and useful as possible. Stop expanding description when it is not useful to expand it further.

Answer to Review Question 5

There are many possible answers to this. The following is just one example.

- 0 process orders
  - 1 process one order
    - 1.1 receive order
      - 1.1.1 note pizza requirements

        - 1.1.1.1 note toppings required
        - 1.1.1.2 note price
        - 1.1.1.3 add to bill
      - 1.1.2 note additional requirements (e.g. drinks)
      - 1.1.3 note name and address
      - 1.1.4 take payment details
    - 1.2 make pizzas
      - 1.2.1 make a pizza to correspond to one order
        - 1.2.1.1 prepare dough
        - 1.2.1.2 add toppings as specified
      - 1.2.2 put pizzas in oven
      - 1.2.3 wait until pizzas cooked
      - 1.2.4 remove pizzas from oven
      - 1.2.5 put each pizza in a box
    - 1.3 deliver pizzas
      - 1.3.1 give pizzas corresponding to one order to delivery motorcyclist
      - 1.3.2 give bill and address to motorcyclist
      - 1.3.3 wait
      - 1.3.4 check payment received against bill and clear account
- Plan 0: do 1 repeatedly (possibly in parallel)
- Plan 1: do 1.1 – 1.2 – 1.3
- Plan 1.1: do 1.1.1, 1.1.3, 1.1.3 and 1.1.4 in any order
- Plan 1.1.1: do 1.1.1.1 – 1.1.1.2 – 1.1.1.3
- Plan 1.2: do 1.2.1 repeatedly until order complete, then 1.2.2 – 1.2.3 – 1.2.4 – 1.2.5
- Plan 1.2.1: do 1.2.1.1 – 1.2.1.2.
- Plan 1.3: do 1.3.1 and 1.3.2 in either order, then 1.3.3 – 1.3.4

Answer to Review Question 6

KB analysis is concerned with hierarchies of objects and actions, and classifies items in terms of relationships such as class membership, attributes, and features (using AND, OR and XOR relationships).

Answer to Review Question 7

True. One way to structure a manual would be in terms of related actions or concepts, and KB analysis would help with this conceptual structuring.

Answer to Review Question 8

Entities include objects (concrete, composite and actors), attributes, actions and events. Note that there are many other terms (e.g. patient, agent, instrument) that are not entities in their own right, but are terms that define a relationship between entities. (If this is not immediately obvious to you, think hard about it).

Answer to Review Question 9

Task analysis focuses on domain entities, whereas ER analysis for database design focuses on the entities that are to be represented in the resulting computer system.

Answer to Review Question 10

Objects have attributes. Actors perform actions. Composite objects comprise (simpler) objects. Objects are located at objects. Actions have patients (objects), actors (objects) and instruments (objects). Actions may take place before events, be triggered by events, or caused by events.

Discussion on Activity 1

In designing your HTA, you will probably have included both searching and browsing as activities. The design consideration is likely to focus on browsing, and on how that can be made efficient and effective.

Discussion on Activity 2

Note that actions should not be described at the level of a button-click, but at the level of something that has some domain significance. Similarly, objects should not be described in terms of icons, but in terms of objects that have some domain significance. Depending on which sites you choose, you may or may not be able to use your informal analysis to propose some design improvements to the sites.

Discussion on Activity 3

As in exercise 4, you should be aiming to identify entities and relationships of various kinds. You should reflect on both what is involved in ER analysis, and what circumstances you would find it most and least useful in.

Chapter 9. Evaluation

Table of Contents

Objectives

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

- Discuss the role of evaluations in interactive system design.
- Describe different methods of evaluation.
- Explain how the results of the different methods can be used to improve designs.
- Decide when it is appropriate to use a method.
- Design useful questionnaires/ surveys for evaluation.
- Design experiments to evaluate usability factors.

Context

In this unit we are concerned with evaluation – how can we assess whether some artefact meets some criteria. Specifically in this unit we are interested in how user interfaces can be evaluated in terms of their usability and how well they meet the clients' requirements. Preece (Jenny Preece et al (1995). Human Computer Interaction.) cites four reasons for doing evaluation:

- To understand the real world and work out whether designs can be improved to fit the workplace better.

- To compare designs – which is best for given situations and users.

- To decide if engineering targets have been met – is the design good enough to meet the specification.

- To decide if the design meets standards.

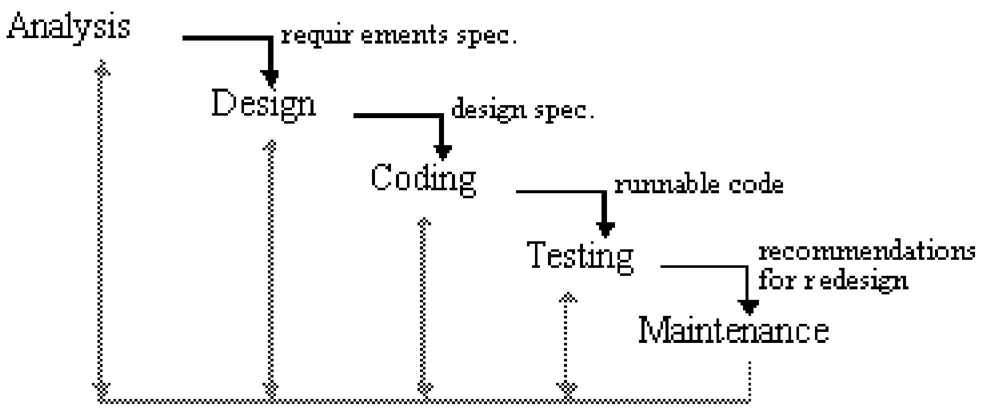
The importance of evaluating usability is evidenced by the myriad of consultancies that have sprung up with the development of the world wide web offering to evaluate web sites for clients. These companies typically produce a report at the end of the evaluation telling their clients what they think is wrong with their web site and possibly what could be done to remedy the situation. Needless to say they also offer web site redesign services. There are two main problems with such approaches to evaluation:

- The client has no way of knowing how the results of the evaluation were arrived at and so can not make an assessment for themselves of the suitability or coverage of the evaluation – whether it looked at the right kind of usability issues, and whether the approach taken was likely to find a good proportion of the problems.

- The evaluation is typically conducted once a lot of time and effort has been put in to designing and building the web site. This may have been wasted if the site needs a complete re-design.

In this unit we look at how evaluation can be performed and so you should gain an insight into resolving the first point – you will be able to perform evaluations and, importantly, understand which evaluations are appropriate for different situations, and what level of coverage can be expected.

The second of the above points requires us to take a step back and think about when evaluation is appropriate. Conventionally, software development was formalised into a waterfall model (see Unit 3 for more details of software development techniques). In this model (illustrated in the following diagram) there was a strict flow of activity from stage to stage (with later additions of feedback to previous stages illustrated in grey).



At the first stage of this model analysis of tasks (see Unit 8 for details of task analysis techniques), users, and the domain are carried out to produce a requirements specification. This document details what tasks the analyst believes are important, what kinds of users perform such tasks, and details of the domain in which the tasks are to be carried out. This document passes to the following design stage where it is used to inform the design of data structures, system architectures, and user interfaces. These designs are described in a design specification which feeds into the coding stage which in turn produces runnable code. Finally this runnable code can be tested to see if it meets the requirements set out in the requirements and design specification documents. In particular, the functionality (whether the code provides the appropriate support for user actions), usability and efficiency of the code are tested. This may then provide some recommendations for redesign which feed back into previous stages of the process and into the maintenance program once the software has been deployed. In some ways this approach relates to the example briefly introduced above – the web site development company designs and codes their web site and then uses outside consultants to test the site's usability.

It all seems so neat and rational doesn't it? That neat and rational appearance is one of the waterfall model's main weaknesses – real world projects are simply not neat and rational and it is difficult to follow such a simple sequence in any realistically sized project. Furthermore, the simple linear sequence of stages from analysis to testing creates its own problems. For a start, it is often hard to completely analyse requirements at the start of the project, especially user interface requirements – more requirements come to light as the project develops. Then there is the issue of runnable code. No code is developed until quite late in the project development cycle which means that any errors in the analysis and design stages go undetected until late in the project (when the damage has been done you might say). This compounds the problem that testing is not done until the end of the project which means that mistakes in analysis, design, and coding are then costly and difficult to repair.

Using the waterfall approach can be costly in terms of redesign, but, you may ask, surely this is unavoidable. For instance, when you visit the hair dressers for a hair cut, the hair dresser usually finds out what you want first (analysis stage), and then sets about designing and implementing the hair cut i.e. they cut your hair. A fascinating aspect of having your hair cut is that there is hardly any way of knowing what the result will look like until the very end when it may be too late. But imagine if the hair dresser had some means of showing you exactly how you would look at the end of the haircut (rather than some faded photograph of a nice haircut from the 1980s), and moreover, was able to show you how the haircut was progressing with respect to your requirements. You would then have a better idea of what the end product would be, and could even modify the design during its implementation.

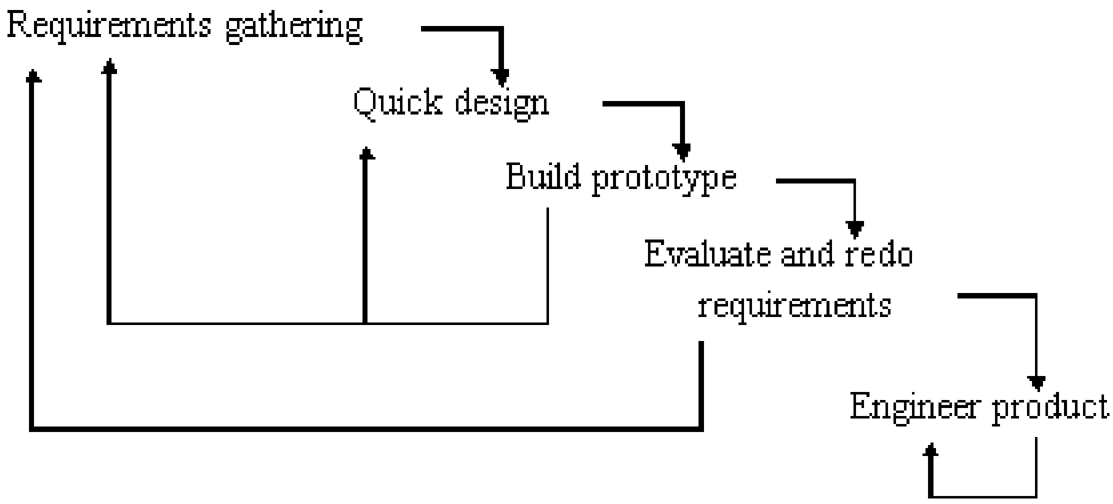
So, rather than evaluating just at the end of the development cycle it makes more sense to evaluate and test designs and products throughout the cycle. One example of this approach is referred to as iterative prototyping (see Unit 3). The following figure illustrates this more iterative and reflective approach to development. As before the approach starts with finding out what the user wants – but at a more general level, what the objectives are etc. These requirements then feed into some quick design process which typically concentrates on the user interface aspects of the system. Design ideas then feed into the development of one or more prototypes of the system – rough and ready systems which exhibit some of the desired characteristics. Even at this early stage feedback may inform the design and requirements stages. The prototypes are then typically evaluated in some way in order to refine the requirements. At this stage not only the customers, but also designers and potential users have their say. This iterative development of the requirements continues until both customer and designer are satisfied that they mutually agree and understand the requirements. As before, these requirements are then used to engineer the product – note that prototypes may also be included in the requirements. There are several advantages to this approach to software development including:

- Feedback on the design can be obtained quickly – rather than creating a complete working user interface. This saves both time and money.

- It allows greater flexibility in the design process as each design is not necessarily worked up into a full working system. This allows experimentation with different designs and comparison between them.

- Problems with the interaction can be identified and fixed before the underlying code is written. Again, this saves time and effort.

- The design is kept centred on the user, not the programming – instead of primarily worrying about coding issues it is the usability of the interface that is of prime importance.



We can see in these two approaches two different times at which evaluation can be applied. First off there are formative evaluations which occur when the design is being formed – e.g. at the requirements and design stage. Such evaluations help inform the design and reduce the amount of costly redesign that may be required after the product has been completed. As such evaluation occurs early in the process it typically evaluates designs and prototypes as opposed to finished products. In contrast there are summative evaluations which occur after the product has been completed – they essentially sum up the product and are usually intended to confirm that the finished product meets the agreed requirements. As these evaluations involve finished products the errors found are likely to be costly to repair.

Activity 1 – Benefits of Waterfall Model

The waterfall model presented at the start of this unit was criticised for being too rigid and not allowing evaluation until the final product had been developed. What advantages do you think the approach could have for software design and development?

A discussion on this activity can be found at the end of the chapter.

Activity 2 Other Models of Software Development

The start of this unit discussed two approaches to software development - the waterfall model and iterative prototyping. There are many more than these two approaches - find another approach and contrast it with those presented here. Pay special attention to the role of evaluation in the approaches you find.

A discussion on this activity can be found at the end of the chapter.

Approaches to Evaluation

Needless to say there are many different ways to evaluate interactive systems. We will cover several representative approaches in this unit, but it is important to remember that these are just a small selection of the many approaches that have been developed. Evaluation itself can involve many different people from designers and evaluators through to actual users of the system. It is important to be clear that in an evaluation there are people conducting the evaluation (evaluators) and usually people participating in the evaluation (subjects) - potential users, evaluators, or even in some cases no-one. In order to give some idea of the variety of techniques available we shall consider evaluation in terms of the kinds of people participating in the evaluation – from users to evaluators, and finally to techniques that do not rely on anyone physically using the system.

Evaluating with Users

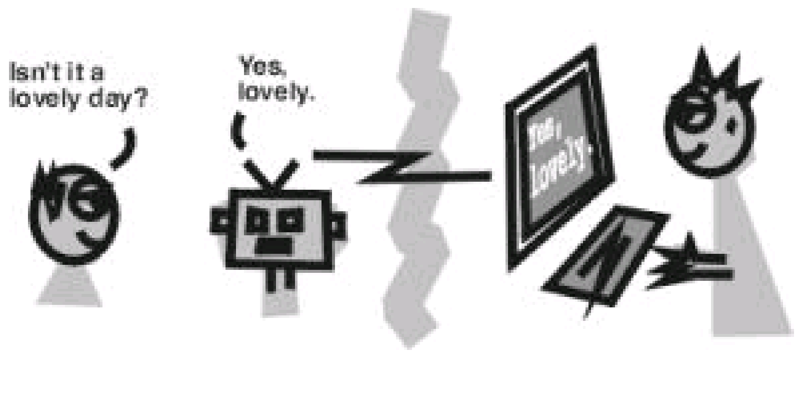
We are interested in evaluating systems which will be used by users, so why not test the systems with people who are likely to use them? Such evaluations are the focus of this section.

User Observation

In order to discover usability problems with user interfaces we can observe users attempting to use them – in which case the users are the subjects of the observation. Typically users are given tasks to perform with the user interface (these are the tasks the user interface is intended to support) and then asked to attempt to complete them. Whilst using the system the users are observed through a number of various techniques as outlined below. It is important to remember that we are interested in problems the subjects find with user interfaces rather than testing the subjects' abilities. In fact it is usually a good idea to tell subjects this before they start the test so that they realise that it is the interface that is being tested, not them.

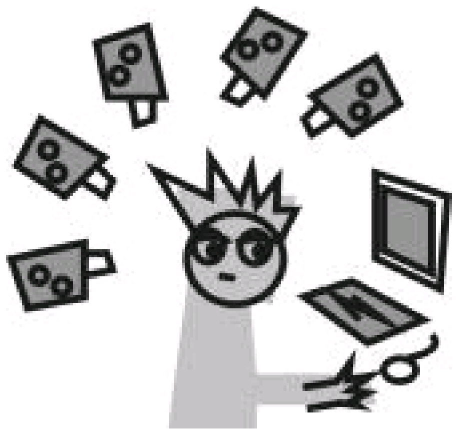
As the idea behind this approach is to see how potential users will react to a system it is important to select appropriate users as subjects – they should be typical of the target population for the system. For instance, evaluations of cash machines will require any subjects who have, or could have, a bank account. Evaluations of equipment to monitor patients' conditions in hospital will require a more select set of subjects – those with appropriate medical training e.g. specialist nurses. Subject selection should be done before the evaluation to ensure subjects have the appropriate experience and knowledge. A simple questionnaire (see later discussions) can be used at the start of the evaluation to check that subjects are appropriate.

Firstly users may be directly observed – the evaluator watches what the user does and notes down pertinent points. Clearly the evaluators may miss important information as the user carries out their task. To address this problem sessions are usually video recorded – focusing on the important parts of the activity such as screen display and/ or mouse movements. These video recordings can then be reviewed later and interpreted at length. Several off-the-peg kits have been developed which comprise of multiple video cameras, video and audio mixing facilities, video recorders, tripods etc. These are often referred to as a ‘lab in a bag' or a ‘lab in a box' and can be relatively easily transported to different locations. These video recordings can often be used in conjunction with logs of use of the system generated by the system itself – a time stamped list of the actions the user performed e.g. pressing a button, moving the mouse to a particular position.



Subject « System « Wizard

However, the more complex the observation becomes, the more it intrudes upon the user and may effect their performance so skewing the results of the evaluation. Sophisticated observations often involve the use of one way mirrors to hide the observer and their equipment from the user. Furthermore, the user may be placed in a setting similar to their normal working environment to make them feel more at ease. Alternatively, observations can be carried out in the workplace itself which gives well grounded results, but can often be difficult to set up, especially in busy working environments where the subjects may be interrupted.



"Just relax and act normally"
– observation can be intrusive

Simply observing the user attempting to perform tasks may not tell us enough about the problems they are encountering. Unlike computer generated logs we can not at present easily get a direct log of their actions and intentions. Instead we have to rely on the user telling us what they were doing and why. Two approaches to generating a protocol of the task (description of what the user does and why) are possible:

- **Concurrent protocol** – the user says what they are doing and why whilst they are performing the task.
- **Retrospective protocol** – the user tells use what they did and why after they have finished the event. This may be done in conjunction with a transcript of the task performance, and/ or a video recording which may help trigger the subject's memory.

Choosing which approach to take is not an easy task in itself. For instance, asking the subject to verbalise their every action and intention may distract them from performing the task itself. On the other hand, subjects may not reliably remember what they did or why they did it in a retrospective protocol. Moreover, retrospective protocols rely on subjects being prepared to stay after they have performed the task to further discuss their actions – how many people would be prepared to do that?

User Feedback

After subjects have used a system (observed or not) plenty of useful information about its usability can be obtained from the subjects themselves. Two methods are considered here: interviews, and questionnaires.

Interviews

Observing the use of a system can give us some idea about what happens when the systems are used. However, it does not give us any idea about subjects' perceptions of the system – how they felt using the system, whether they would like to use such a system etc. In interviews we attempt to find out how the subjects felt about the system – their subjective opinions.

Typically an interviewer asks an interviewee a set of questions about their experience of the system. These questions may be structured in several ways:

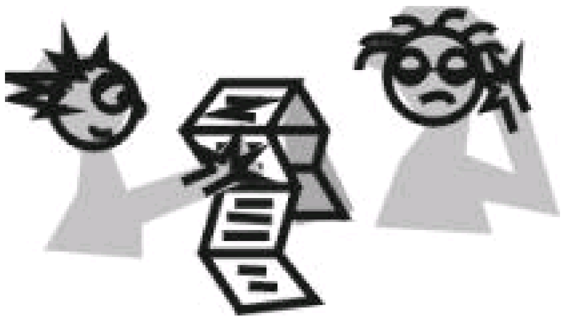
- **Structured interviews** – the questions are defined beforehand and are the only questions asked during the interview. This style of interview does not allow exploration of subjects' opinions, but does provide a set of answers which can be compared across several subjects to derive some generalisations of peoples' opinions of the system.

- **Unstructured interviews** – some set of topics of interest in to the evaluation are defined, but the interviewer is free to choose which topics they ask, in which order, and how much they explore the topics. This approach usually does not give as easily comparable results as structured interviews, but it can be useful in identifying requirements – these are useful situations in which the full evaluation criteria are probably not well defined yet.

- **Semi-structured interviews** – these combine the structure of the structured interview's predetermined set of questions with the flexibility to pursue topics further if they seem interesting.

- **Card sorting** – this technique is quite different from the others. Instead of being asked questions the interviewee is given a set of paper cards with some aspects of the user interface or task written on them. They are then asked to sort the cards into groups – typically groups of similar items. In this way the interviewer can get an idea of interviewees' perceptions of the nature of the user interface and/ or the task. This perception might be quite different to what the designers envisaged.

When designing the interview it is important to remember that the interviewee's time is probably quite restricted, so the length and depth of the interview should carefully constrained – this will also save the interviewer from spending a substantial amount of time conducting the interviews as well as later transcription and analysis time.



"Now to question 149" –
try not to tire the subjects

Questionnaires

Unlike interviews, questionnaires involve subjects filling in responses to questions themselves rather than responding to an interviewer. Therefore questionnaires can typically involve a larger set of people as there is no need for the interviewer to ask each subject for their answers. Indeed, extensive use of questionnaires is usually referred to as a survey. As with interviews, questionnaires need to avoid being too long and time consuming. In addition they need to be designed to ensure enough subject response. As a rule of thumb, questionnaires should be kept within two sides of A4, and some sort of incentive will probably need to be given to encourage potential subjects to complete it. There are two main types of questionnaire:

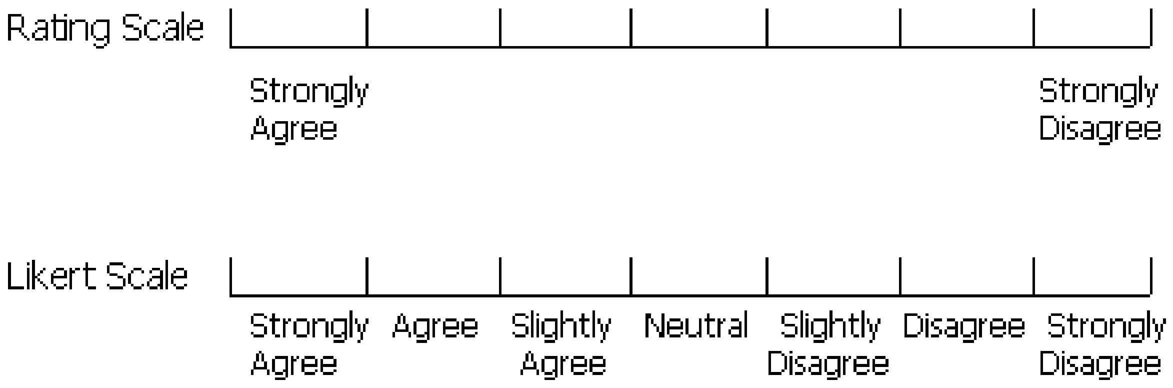
- Open questions – the subject is free to write their answers in any way they see fit. Of course, this means that analysis time must be devoted to trying to understand what subjects meant by their answers, and moreover, subjects may not provide appropriate answers at all.

- Closed questions – the subject selects an answer from a set of presented possibilities. This provides less ambiguous results than open questions and can even be used to provide some numerical results such as ‘nine out of ten cat owners, who expressed a preference, said that their cat preferred Super-Sweet cat food'.

- Closed question questionnaires rely on some way for the subject to select between alternative possible responses. Preece (1995) identifies several kinds of scale from which subjects can choose their response including:

  - Simple checklist – simple responses such as ‘yes', ‘no', or ‘don't know' are provided.

  - Multi-point rating scale – a number of points on a scale are provided which give a range of responses e.g. ranging from ‘Strongly agree' to ‘Strongly disagree' as illustrated below. A variation of this is the Likert scale (illustrated below) which indicates the meaning of each point e.g. instead of simply putting ‘strongly agree' at one end of the scale and ‘strongly disagree' at the other, the intermediate points of ‘agree', ‘slightly agree', ‘neutral', ‘slightly disagree', and ‘disagree' are included.



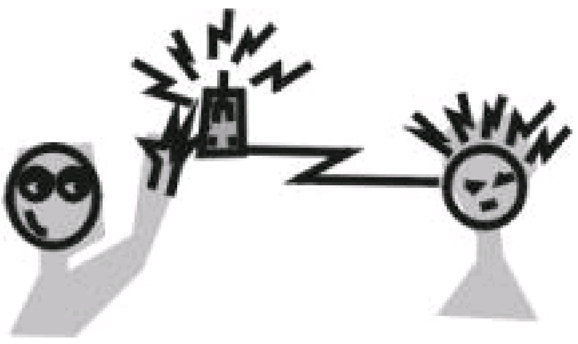
  - Ranked order – rather than giving the subject a scale from which to pick their responses, this approach requires the use to specify their preference for items in a list. For example, a subject may be asked to rank (using the numbers one to four) their perception of the usefulness of four different commands.

Summary of User Feedback

We have looked at two main ways to get feedback from users about user interfaces – interviews and questionnaires. They both have their strengths and weaknesses, and give different kinds of results. Next we will look at how potential users can be utilised to provide more rigorous results than we could get from the user observation discussed previously.

User Experimentation

Of course, we don't mean experimenting on users, but rather performing experiments in which subjects participate. Instead of simply giving users tasks and observing how they perform them, experimentation is concerned with attempting to provide some empirical evidence to support some hypothesis. In the case of user interfaces, a hypothesis is usually some supposition that one user interface will be better than another in some respect.



"This won't hurt a bit"
- Experiments on users

Essentially an experiment involves two sets of users whose situation differs in only one respect – that which is being tested. For example, in an experiment to determine whether a trackpad is quicker to use than a mouse for selecting objects on the screen the two sets of subjects would be treated identically except that one set would use a trackpad and the other a mouse. In this example experiment the hypothesis might be that 'using a mouse is quicker than using a trackpad for selecting objects on the screen'.

This simple example highlights many aspects of experimental design. First off, the hypothesis is some prediction we wish to test (here that a mouse will be quicker to use than a trackpad) – this has itself probably been derived from some previous research or observation. Secondly, we need to be explicit about what kind of difference we expect to see – this is the thing we are going to measure in the experiment and is referred to as the dependant variable. In this case it is the time to complete tasks with the computer (the prediction was that using a mouse would be quicker than using a trackpad). Thirdly, we need to state what we are going to change between the different sets of subjects – the thing that we change is referred to as the independent variable. In this case it is the kind of input device used – either a mouse or a trackpad. Usually in experiments to do with user interfaces we keep the task the same across all subjects so that we can determine the difference between user interfaces for the same task – therefore it is important to select tasks which can be performed using the different user interfaces. Finally, it is important in experiments to avoid confounding variables. These are things which are beyond your control and might effect the result of the experiment and should therefore be avoided. In this case the fact that most users of computers currently use mice rather than trackpads could be a confounding variable as the subjects would be more experienced with the mouse and so probably quicker by default. In order to remove this confounding variable we could select only novice computer users for the experiment i.e. those without any experience of using computers and therefore, we might assume, mice.

Independent Variables

As mentioned previously, independent variables are the factors that we change between sets of subjects. Previously we considered the type of input as an independent variable – either mouse or trackpad, these are levels of the independent variable. In this example there are two levels of the independent variable, but more complex experiments may involve more than two levels of independent variable, and may also involve more than one independent variable. Each unique combination of independent variable levels is referred to as a condition. So, for our simple example there are two conditions in the experiment – one where subjects use a trackpad, and another where they use a mouse. If we were to build on the design of this simple experiment by including another independent variable we would increase the number of conditions in the experiment. For example, we may want to change the hypothesis to be that 'using a mouse is quicker than using a trackpad, and providing on-line help makes them easier to learn to use'. In this case we would have an extra independent variable – the kind of help provided. We might wish to have three levels of this independent variable – no help, simple textual help, and interactive help. The following table illustrates how these two independent variables and their levels combine to produce six conditions in the experiment.

	Input with Mouse	**Input with Trackpad**
No help	Condition 1: No help, mouse input	Condition 2: No help, trackpad input
Simple Text Help	Condition 3: Simple help, mouse input	Condition 4: Simple help, trackpad input
Interactive Help	Condition 5: Interactive help, mouse input	Condition 6: Interactive help, trackpad input

Dependent Variables

Dependant variables are those things that we are trying to measure to support or disprove our hypothesis. Therefore they need to be things that are easily observed and measured, for example:

- The time it takes to complete a task
- The number of errors made during the experiment
- The time taken to recover from an error
- Number of times help is used
- Length of time help is used for
- Time spent before responses formulated

The dependant variables above produce quantitative data – i.e. numerical results. We might also use qualitative data such as subjects' preferences – how much they liked the system, or the quality of product produced at the end of the experiment. Qualitative data such as user preference can be gathered using interviews or questionnaires as discussed previously.

Assigning Subjects

Recruiting enough subjects is one of the hardest parts of conducting an experiment. Typically at least ten subjects are needed per condition to provide statistically significant results – results that we are pretty sure could not have happened by chance. For the more complex experiment proposed previously this would require sixty subjects (ten per condition) – attracting so many potential subjects usually requires some incentives such as cash or unit credits for students. The use of each subject in only one condition is referred to as between-groups design. We can reduce the number of subjects needed by reusing subjects in different conditions – referred to as within-groups design. However, subjects may then carry over experience from one condition to another and so introduce a confounding variable into the experiment.

Supposing we were setting up the simple experiment described above which attempts to show that using a mouse is quicker than using a trackpad. The independent variable in this case is the type of input, with two levels – mouse or trackpad. There are therefore two conditions – 1) mouse, 2) trackpad. Suppose we manage to find ten subjects for the experiment: Ann, Bert, Carol, Derek, Eric, Fred, Gary, Harry, Iris, and Joan, then we can assign the subjects as follows:

Table 9.1. Between-groups design

Mouse input:	Ann	Bert	Carol	Derek	Eric
Trackpad input:	Fred	Gary	Harry	Iris	Joan

Table 9.2. Within-groups design

Mouse input:	Ann	Bert	Carol	Derek	Eric

Trackpad input:	Ann	Bert	Carol	Derek	Eric

Statistics

We use statistics to determine whether an independent variable (such as the kind of input device) has an effect (positive or negative) on a dependant variable (such as the time to complete a task), and that this has not come about by chance. A thorough coverage of statistical analysis is outside the scope of this unit – suffice to say that we are looking for results that are statistically significant, that is, we believe that they did not occur purely by chance.

Summary of User Experimentation

To sum up, performing an experiment involves the following stages:

- Formulate hypothesis – some statement that you wish to test in the experiment
- Identify the variables – independent, dependant, and confounding
- Decide on the levels for the variables
- Select subjects – decide who would make suitable subjects
- Decide on the experimental design – within-groups or between-groups
- Decide what tasks are to be performed
- Perform the experiment
- Analyse the results

Summary of Evaluating with Users

This part of the unit has looked at how evaluation can be undertaken using potential users of the system. The fact that we are using potential users to test the system means that we can feel pretty confident that if the user group involved in the evaluation finds it easy to use then so will real users of the system. However, this might be just for the limited task(s) we asked the users to perform. Average users are not experts on evaluation and so would not be able to identify possible additional problems. Moreover, evaluations involving users take a long time. In our simple experiment involving users we needed sixty subjects to test two factors of the interface design – kind of input, and kind of help. Each test of the system may take at least an hour, and then the analysis time must also be taken into account. The next sections discuss alternatives to user testing which have their own pros and cons.

Activity 3 – Experimental and Questionnaire Design

Design an experiment to determine which sounds users prefer to use as email arrival notification (e.g. a simple beep, a duck quaking, someone coughing, a trumpet etc.). Extend this experiment to consider what visual notification they prefer - none, a flashing icon, or a dialogue box. Think about the hypotheses and variables, and suitable people to act as subjects.

Try to address the same questions using a questionnaire - what audio and visual indication do people prefer when email arrives. Consider how you could make the questionnaire more general so that any member of the public could complete it. Also, discuss how this would feed into the design of email programs.

A discussion on this activity can be found at the end of the chapter.

Activity 4 – Comparing User Evaluation Techniques

Compare the use of experimental studies to user based observation. Discuss the differences between the approaches and the different kinds of information that can be acquired from them. Also consider the different situations in which they are appropriate.

A discussion on this activity can be found at the end of the chapter.

Review Question 1

What are the differences between concurrent and retrospective protocols involved in user observation? What effect do these differences have on user observation?

Answer at the end of the chapter.

Review Question 2

In an experiment to determine which user interface (command-line, desktop metaphor, or virtual environment) allows users to perform their task quickest, what might the following be:

- The independent variables and their levels
- The dependent variables
- The conditions
- The confounding variables

Answer at the end of the chapter.

Evaluating with Evaluators

Evaluating systems with potential users means that we can get some idea of how it would be received by people when they use it. However, potential users have not typically been trained to identify usability problems. In this section we look at the use of trained evaluators in the evaluation of systems.

Heuristic Evaluation

In order to address the problem of costly evaluations involving tens or hundreds of users, Molich and Nielsen (1990) devised a technique called heuristic evaluation. In their approach a heuristic is a usability principle against which the system is evaluated by trained evaluators. They initially considered ten heuristics, but later went on to refine them further.

In an evaluation there are typically three to five evaluators who each assess the system (or a prototype of it) against the ten heuristics. Before performing the evaluation all the evaluators will need to be trained in terms of the evaluation technique, and the tasks that the system is supposed to support. The evaluation itself usually involves working through a scenario of use (an example task that users might perform with the system). In order to identify problems thoroughly the evaluators usually work through the scenario twice – once to get an overview of the system, and the second time to assess the usability in depth. After they have completed the evaluation the lists of problems found are compiled into a coherent set – using multiple evaluators means that there is more chance of identifying a coherent set of problems as each evaluator probably perceives problems slightly differently. The problems in the collated set are then rated according to a severity scale (discussed later). Finally, the problems found are fed back into design to improve the next design of the system. As discussed earlier in the unit, this form of iterative prototyping and evaluation works best if the evaluation occurs early in the development process – before too much coding time has been invested in the system.

Heuristics

The first set of heuristics developed for heuristic evaluation are listed below. Typically the evaluator would work through the given scenario trying to decide whether the heuristics were violated or not.

1. **simple and natural dialogue** – is the interaction with the system clear and concise?

2. **speak the users' language** – can the user understand the information presented to them? E.g. is there too much jargon?

3. **minimise users' memory load** – does the system rely on users remembering complex or meaningless pieces of information? E.g. pressing ctrl+alt+del is a pretty meaningless way to bring up the task manager for a novice user. Another example might be a system in which users cannot copy information from one window to another. Therefore they have to memorise the information and retype it. This is a large burden for users' memory, instead the system should be developed to support such copying (maybe it wasn't though of in the initial design – another reason for more iterative prototyping).

4. **consistency** – do things happen differently in different parts of the system? E.g. the operation to save a file should be named consistently in different parts of the system, not save in one part and write in another.

5. **provide feedback** – does the system inform the user about what is happening? E.g. highlighting buttons when pressed is good feedback, as is showing an hourglass when the system is busy.

6. **clearly marked exits** – can the user easily abort an operation? E.g. in wizards there should be some way of escaping from the sequence determined by the wizard – and it should be obvious how to do it.

7. **provide shortcuts** – if the user becomes an expert, will they be able to use shortcuts to speed up their performance? E.g. providing keyboard based shortcuts for menu items.

8. **precise and constructive error messages** – just telling the user that there was an error does not help them to understand the system. Does the system explain why there was an error and what could be done about it?

9. **prevent errors** – if possible, does the system prevent users from making errors? E.g. if a field requires only numerical data, does the system prevent the user from entering letters?

10. **adequate help and documentation** – could the user work out how to use the system from the help and documentation provided?

Severity Ratings

As we mentioned before, once the usability problems have been identified using the heuristics, the severity of the problem is rated. Rating is done individually by evaluators and then, as with the problems, the ratings are collated to get the consensus opinion. The rating itself is a combination of the frequency of the problem, its impact, and how persistent the problem is (is it a one-off, or does it happen each time) and is a number from zero to four on the following scale:

- 0 - don't agree that it is a usability problem
- 1 - it's a cosmetic problem
- 2 - minor usability problem
- 3 - major usability problem - important to fix
- 4 - usability catastrophe - imperative to fix

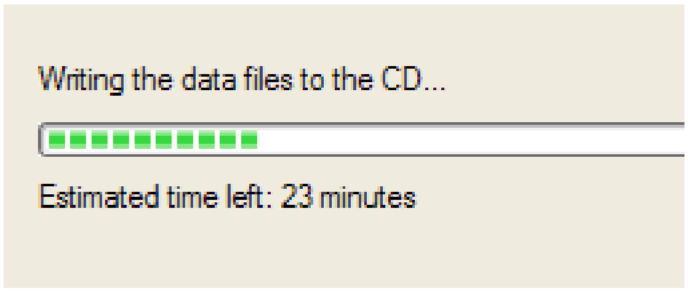
From looking at this scale you will probably realise that it is often hard for evaluators to decide which rating is appropriate.

Revised Heuristics

As mentioned previously, the initial set of heuristics were refined after extensive use. These refined heuristics are discussed below. Note how some of the heuristics have remained unchanged (e.g. 10:

‘help and documentation') whereas others have changed completely (e.g. 1: ‘simple and natural dialog' is no longer present) – why do you think that is?

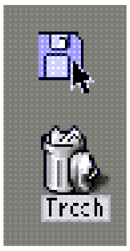
1. visibility of system status – does the system keep the user informed about what is going on? E.g. indications of response time. As a general rule, for response time, if it is less than 0.1 seconds then no special indicators are needed. However, after about 1 second users tend to loose track of what is going on, indeed, 10 seconds is roughly the maximum duration a user will stay focused on one action. For longer delays than this a percentage-done progress bar should be used to indicate what the system is up to e.g. whilst writing a CD, the software shows the following dialog which includes a progress bar, indication of what it is currently doing, and expected time remaining to complete the activity.



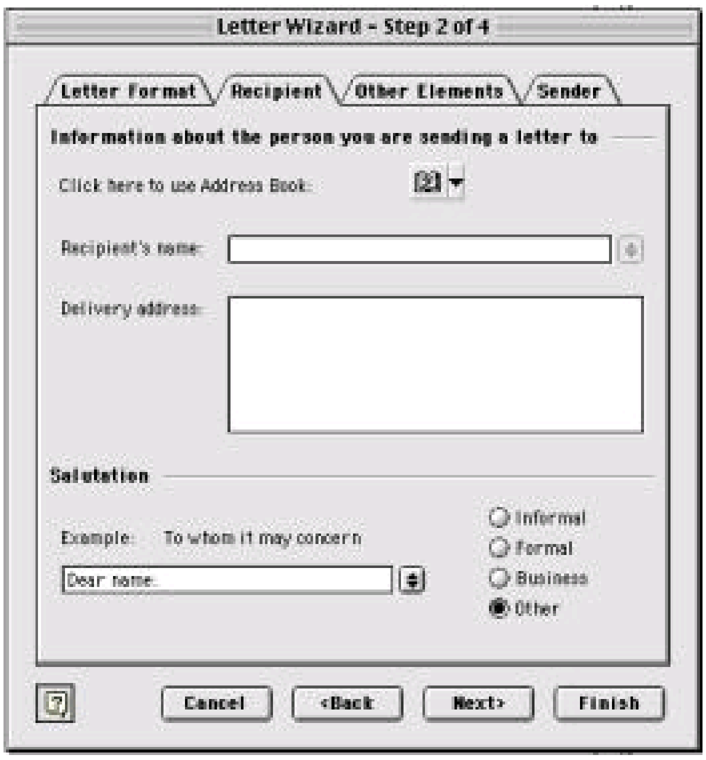
Writing the data files to the CD...

Estimated time left: 23 minutes

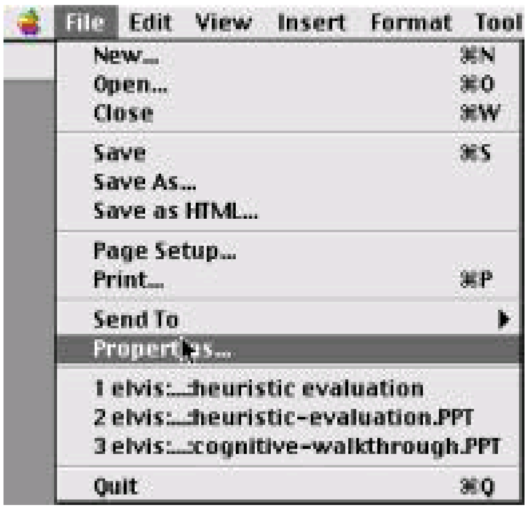
2. match between system and real world – does the system speak the user's language (original heuristic 1.2) and follow real world conventions? A classic example of this heuristic being flouted is the way in which discs are ejected from MacOS based computers. Usually the ‘trash' on the desktop of a MacOS is used to delete files i.e. to throw them away – matching the real world. However, to eject a floppy disc it must be put into the ‘trash' which breaks the match between the system and the real world (it is also inconsistent – revised heuristic 2.4) as illustrated below.



3. user control and freedom – the user shouldn't be forced down fixed paths - does the user have ‘exits' for mistaken choices? E.g. undo and redo. e.g. wizards (as illustrated below) – users are forced down fixed paths to complete tasks (note the <Back and Next> buttons which dictate the sequence; note also the Cancel and Finish buttons which provides some level of user freedom), this is OK for uncommon tasks such as configuring a modem, but not good for frequent tasks which should have a different style of interaction.



4. consistency & standards – as well as the consistency within the system as discussed in the previous set of heuristics, we also need to consider consistency between applications. E.g. in Mac OS X there are consistent menu names and shortcuts between different applications – ‘New', ‘Open', and ‘Close' are all consistent across applications, as are their shortcuts (illustrated in the following picture).

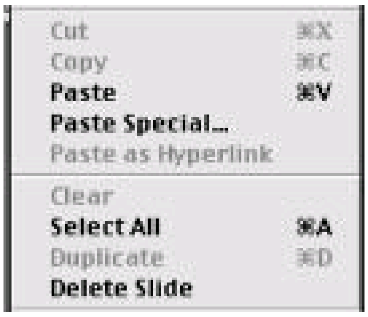


Note also in the above illustration that all the words are consistent in terms of their typography – which letters are capitalised and which are not. Mixing capitals and lower case words inconsistently may cause confusion in users, but would probably not be spotted during the user observations discussed earlier in the unit.

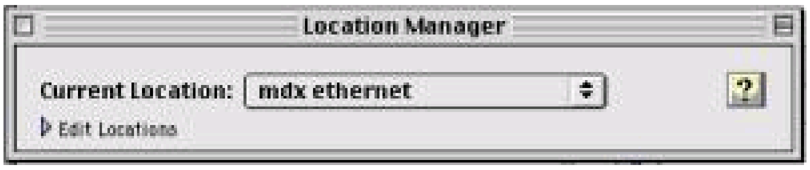
5. error prevention – see original heuristic 9.

6. recognition rather than recall – generally the same as original heuristic 3 (minimise memory load).

7. flexibility and efficiency of use – does the system support shortcuts for experts (illustrated by the shortcuts provided in the following MacOS menu) and is it possible to tailor the system? E.g. providing support for macros to automate frequent tasks.



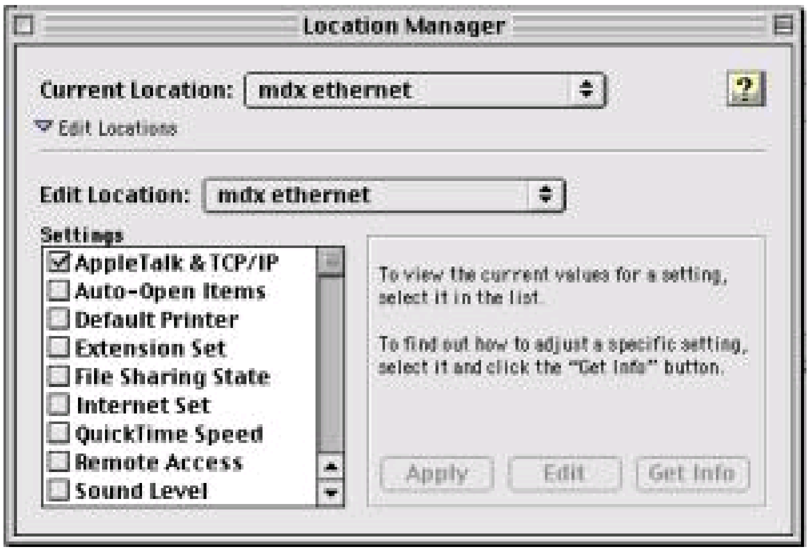
8. aesthetic and minimalist design – does the system present irrelevant information? E.g. the following Mac OS X illustrates an attempt to hide information from the user when it is not relevant – the first dialogue below shows the minimal information needed, but can be expanded to the second interface if the details need to be viewed and edited.


Location Manager

Current Location: mdx ethernet

Edit Locations

Minimalist


Location Manager

Current Location: mdx ethernet

Edit Locations

Edit Location: mdx ethernet

Settings

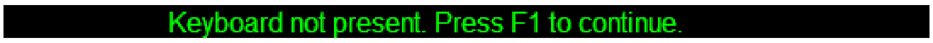
- [x] AppleTalk & TCP/IP
- [ ] Auto-Open Items
- [ ] Default Printer
- [ ] Extension Set
- [ ] File Sharing State
- [ ] Internet Set
- [ ] QuickTime Speed
- [ ] Remote Access
- [ ] Sound Level

To view the current values for a setting, select it in the list.

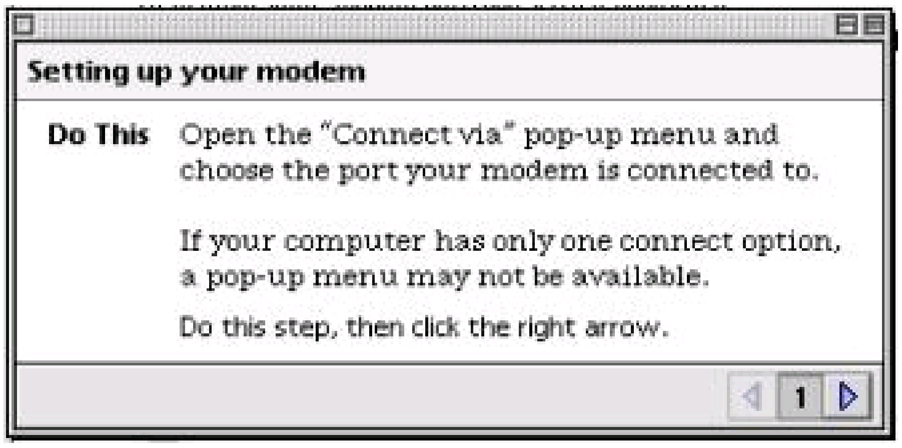
To find out how to adjust a specific setting, select it and click the "Get Info" button.

Extra Detail

9. help users recognise and recover from errors – are the error messages in plain language? Do they precisely indicate the problem, and importantly, constructively suggest a solution? E.g. in the past, command-line based systems have been known to give errors such as the following – this tells the user the problem, but the solution is hardly constructive.


```
Keyboard not present. Press F1 to continue.
```

10. help and documentation – is the information provided easy to search, focused on the user's task, and not too large? Moreover, does it list concrete steps to carry out? E.g. Mac OS X provides constructive help on setting up the modem for you computer as illustrated below - there is a list of concrete steps which relate to the task at hand and are not too long. In addition there is some indication on the actual interface in question and of what operations need to be carried out so reinforcing the help (indicated by the red ellipse on the modem control panel below).



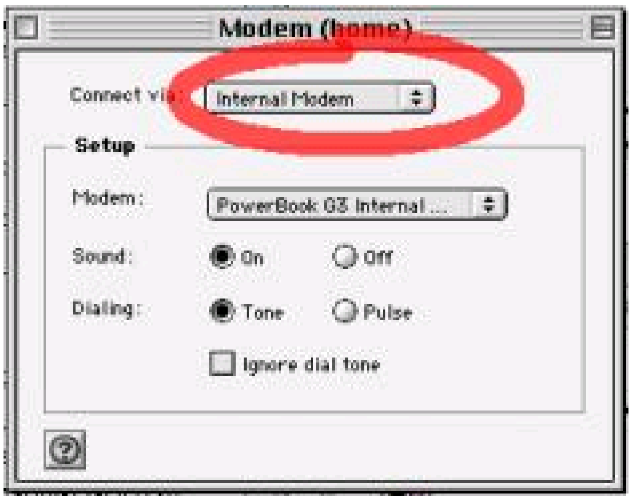
Setting up your modem

Do This Open the "Connect via" pop-up menu and choose the port your modem is connected to.

If your computer has only one connect option, a pop-up menu may not be available.

Do this step, then click the right arrow.

Clear list of steps, and not too long



Modem (home)

Connect via: Internal Modem

Setup

Modem: PowerBook G3 Internal ...

Sound: [✓ On] Off

Dialing: [✓ Tone] Pulse

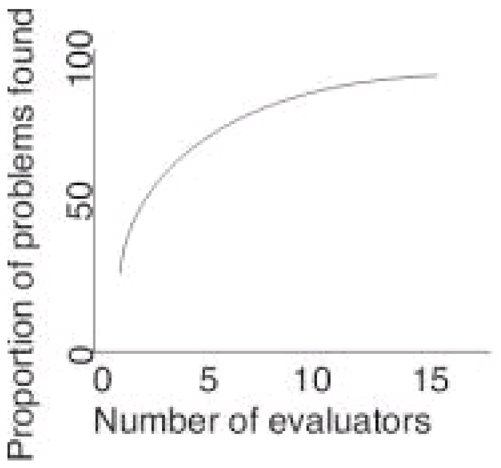
Ignore dial tone

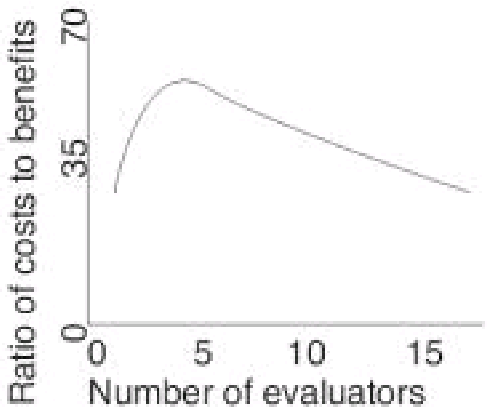
Additional help on the interface itself

Heuristic Evaluation in Use

So far we've considered what the heuristics are, and given some examples of interfaces meeting or violating them. Here we give an example of how the severity might be assessed. Consider the example given in the original heuristics 1.4 (Consistency). In that example the interface used the string Save on the first screen for saving the user's file but used the string Write file on the second screen. Users may be confused by this different terminology for the same function. By a process of elimination we can arrive at a severity rating for this example. First, it is a problem, so that rules rating 0 out. Then, its more than a cosmetic problem (just something to do with the prettiness of the interface) ruling out rating 1. It might be a minor or major usability problem, but it is definitely not a usability catastrophe. Herein lies one of the problems with heuristic evaluation which explains why several evaluators are used and then the consensus taken – how can we make such an objective decision about something which may in some ways be subjective i.e. my own feelings about the interface. Personally I would give it a rating of 3 – major usability problem which is important to fix – as I believe that it could cause some real confusion for users and should really be fixed for the next design.

Once again we have touched on the subject of the number of evaluators needed for a heuristic evaluation. Nielsen himself discusses this issue and considers that a single evaluator achieves poor results as they typically only find 35% of usability problems whereas 5 evaluators find around 75% of usability problems. This raises the suggestion of just throwing as many evaluators as possible at the problem. However, Nielsen argues that many more evaluators won't find proportionally more problems as illustrated in the first graph below. Moreover, as each evaluator costs money to employ the cost benefit ratio decreases rapidly after five evaluators as illustrated in the second graph below.





Cognitive Walkthrough

Compared to heuristic evaluation, cognitive walkthroughs are less prescriptive in terms of the kinds of problems that should be looked for in the evaluation. The emphasis is more on learnability and the basic approach is that the evaluators step through a scenario using the system and at each step attempt to decide whether users would have trouble in moving to the next – hence the term walkthrough.

The first stage of a cognitive walkthrough is to define the inputs to the evaluation - what things we are going to consider in the evaluation. The first input we need to consider is the users. In particular, who are the typical users going to be, and what experience and knowledge can we expect them to have. Once we have defined who are potential users are we consider what tasks they will be performing with the system. The descriptions of the tasks to be supported often come from the Task Analysis stage (see Unit 8). Once we have understood the tasks we need to determine the action sequences of the tasks. That is, we need to work out what each step involved in completing the tasks are, and what sequence they are in. This description of the task action sequence will constitute the steps that are walked through by evaluators later. Finally, we need a description of the system itself – in particular its user interfaces. This may be an implementation of the interface, a prototype, or some mock up drawn on paper, just so long as the task can be stepped through using the description given.

Once the inputs have been defined we need to select our analysts to perform the evaluation. Typically a developer can fulfil this role with a little training. However, better results are obtained when people with some knowledge of cognition (see Unit 4) are used as they will have a better understanding of users' cognitive limitations.

Now that our inputs have been defined and our analysts selected we can proceed with the evaluation. Each analyst steps through the sequence of task actions that are necessary to complete the task. At each step the following questions are asked:

- Would a user try to achieve the right effect?
- Would a user notice that the correct action is available?
- Would a user associate the action with the desired effect?
- If correct action performed, would progress be apparent?

For each of these questions we need to note down not only the answers, but also additional important information including the knowledge requirements of the user, our assumptions about the user, and possible design changes. This extra information can provide important insights for redesign of the system.

Finally, once we have stepped through the task actions and noted down all the problems and extra information we need to work out how this feeds into redesign of the system. Considering the first question above first, supposing the user fails to try to get the right effect (i.e. perform an action) with the user interface, we need to consider what could be done in the redesign to address that problem. Two possible solutions are simply to remove that action from the task (maybe our task analysis had a mistake in it), or to prompt the user for the action, so making it clear that they are supposed to do something at that point. Alternatively, some other part of the interface could be changed so that the user knows that they should try to perform an action.

Moving on to the second question – whether the user notices that the correct action is available. We need to consider what to do in terms of redesign if the user does not notice that the correct action is available. One possible solution is to make the action more obvious in the interface – maybe it is hidden in some sub menu. Similarly, failing the third question would mean that the user did not know which action is the correct one. In this case we might label parts of the interfaces such as buttons or menu items based on our belief's about potential users' knowledge – what labels would mean that users could differentiate actions.

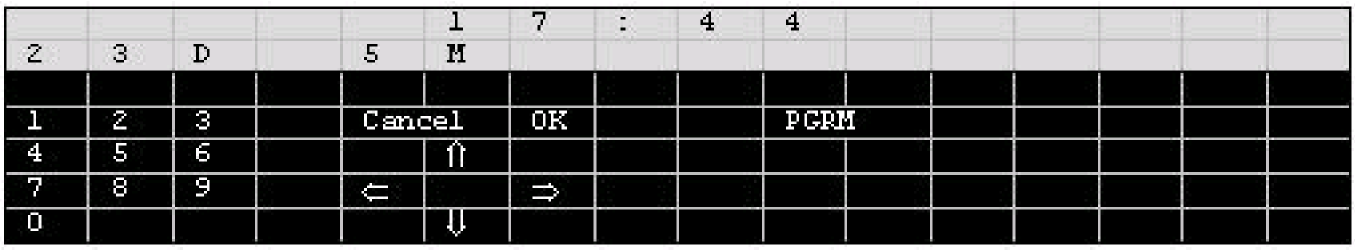
Finally, supposing progress is not apparent when a correct action is performed (the fourth question). This means that the user has no way of knowing if things are going well with their attempt to complete their task. A redesign solution would be to give some feedback to the user, maybe at least an hourglass to indicate that some action is being performed. Better still, the interface can indicate that progress is being made and additionally indicate what it is doing to contribute to this progress (see the example of heuristic 2.1 in the heuristic evaluation section).

Cognitive Walkthrough in Use

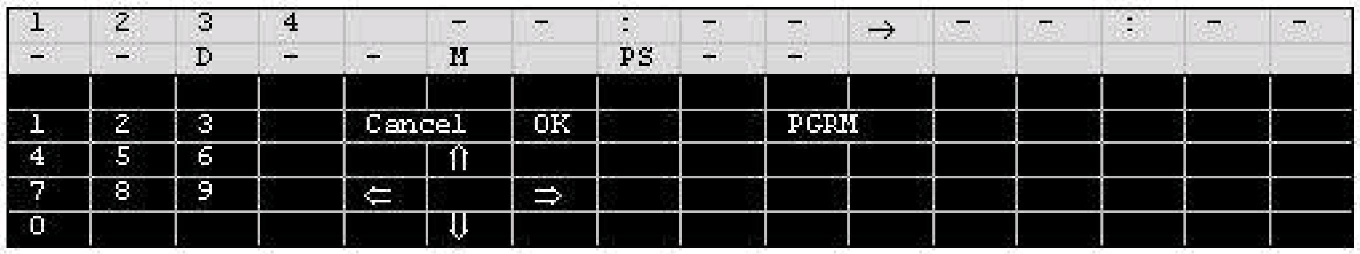
To illustrate the use of cognitive walkthrough, consider the following example of programming a video recorder to record a program, using the remote control.

First we determine the inputs, in this case our potential user group will be anyone with some knowledge of working a video recorder. Next, our task is to program the video to record a program, and we are using the remote control. The specific task instance that we are going to use for the test is to record a program on BBC2 (channel 2 on the video recorder) which starts at 10:30pm and finishes at 11:20pm, and is shown tomorrow. We will work out the action sequence from the description of the interface (another input) that follows.

The remote control handset which we will use to program the video is illustrated below in ‘dormant' mode, that is, when no programming is being performed.

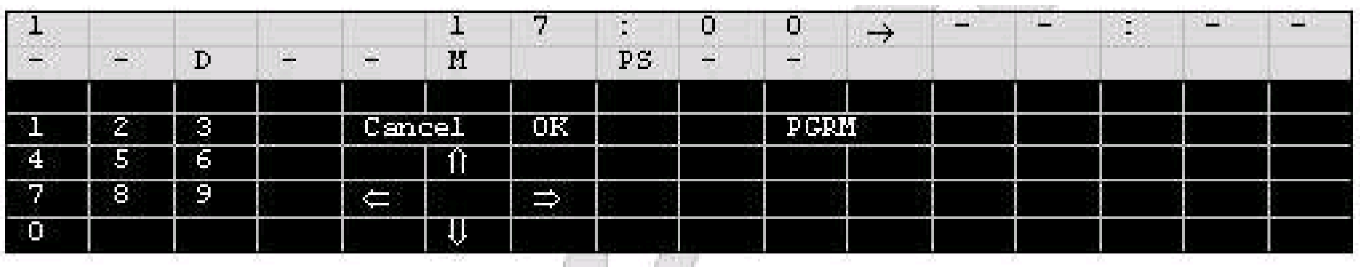


You can see from the interface several groups of objects. The light grey area is the LCD which displays numbers to the users – above this is the current time (17:44) and the date (23/5). The black area contains several buttons which are grouped together into numerical (on the left), navigation (in the middle), and others (to the right of middle). Only one of these other buttons has been shown as it is the only one relevant to the task – the PGRM, or program button. Once this button has been pressed the remote control goes into programming mode. So, if we list the actions of the user and the responses of the remote, our first action is to press PGRM, and the remote's response is to change the display as illustrated below.



For the sake of brevity, the following description only lists the start of the action sequence for programming the video recorder to record our chosen program in terms of user actions and remote control responses.

Action A:	Press PGRM.
Response A:	Display changes to program mode. The digits 1 2 3 4 at the top left of the display flash (see diagram above).
Action B:	Press =>
Response B:	1 stays on, other numbers disappear. 17 starts flashing and 00 is steady next to it as illustrated below.



Action C:	Either press up and down arrows until 17 is replaced by 22, or type 22 on the keypad.
Response C:	22 displayed instead of 17.
Action D:	Press =>
Response D:	00 flashes to the right of 17:

Even with this short action sequence there are plenty of design issues raised. Firstly, in action A, we might wonder whether the user notices that the correct action is available (question 2), or whether they would associate pressing that button with programming the video recorder (question 3). It is not at all clear that the user should press PGRM to set the video up to record. At least when we press PGRM our progress is apparent as the display changes (question 4).

In response A we encounter a display in which four numbers are flashing. This number refers to the number of possible timed video recordings rather than the channel number as user might well expect. This response A means that user might not try to achieve the right effect at that point (question 1) as they attempt to set the channel to be recorded, instead of the number of the timed video recording as the remote control expects. Once the appropriate recording number has been selected (in this example it is 1 by default) the user must try to move on to set the start time of the recording. We can assume that they know that they should do this (question 1), but how they move on is another issue. In order to move on to the next step the user must press => , but maybe they would have thought that OK was the most appropriate button (question 3). Once again, when the correct action has been performed the display changes to flash the number 17 and so we can see that progress is apparent (question 4).

Activity 5 – Heuristic Evaluation

Perform a heuristic evaluation of the remote control example. What usability problems do you find, and how could you improve the user interface.

A discussion on this activity can be found at the end of the chapter.

Activity 6 – Cognitive Walkthrough

Continue the cognitive walkthrough of the remote control example (the last two actions). What extra usability problems do you find and how could the remote control be changed to address all the issues raised.

A discussion on this activity can be found at the end of the chapter.

Review Question 3

Consider the remote control example given in the discussion of Cognitive Walkthroughs. Write down which of the first set of heuristics (from Heuristic Evaluation) you think the remote control as described violates, and which aspects meet these principles.

Answer at the end of the chapter.

Summary of Evaluator Based Evaluation

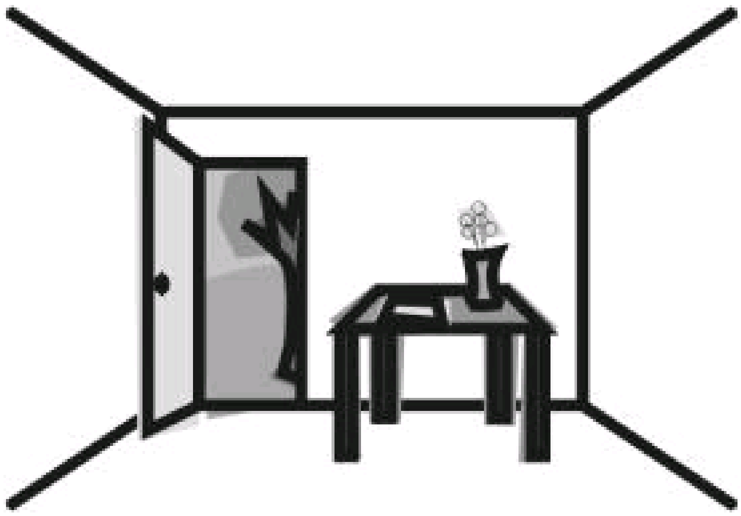
In this section we looked at evaluator based evaluation which differs from user based evaluation in that the people using the systems, or designs, to be evaluated have been trained in identifying usability problems. However, they may not know as much about the task that the system is designed to support as actual users. Moreover, they are not necessarily the kind of people that will use the system in the future. Furthermore, compared to user based evaluation, evaluator based evaluation takes far less time, effort, and participants. It also does not require evaluators to interpret users' actions (from observation) which may not always be accurate. In order to get an all round idea of the usability of a system it is probably a good idea to alternate between user based and evaluator based evaluation. The next section will cover another important form of evaluation which should be included in this set of alternatives.

Model Based Evaluation

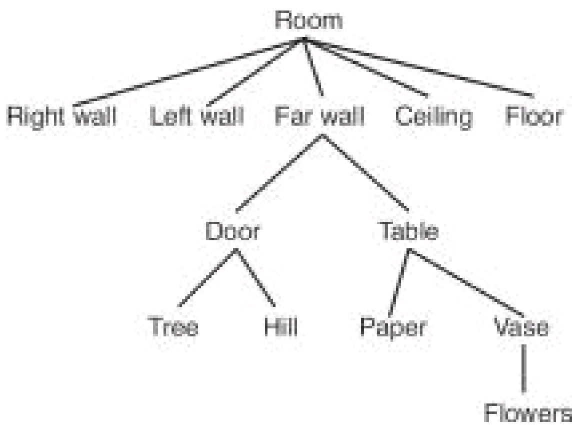
So far we have considered evaluation techniques in which someone (a user or an evaluator) actually uses the interface, or some mock-up of it. Another approach to evaluation covered in this section is the use of models (see Unit 7 for a discussion of modelling) of the interface and understandings of human cognition to evaluate interfaces without actually using them.

Perception Based Evaluation

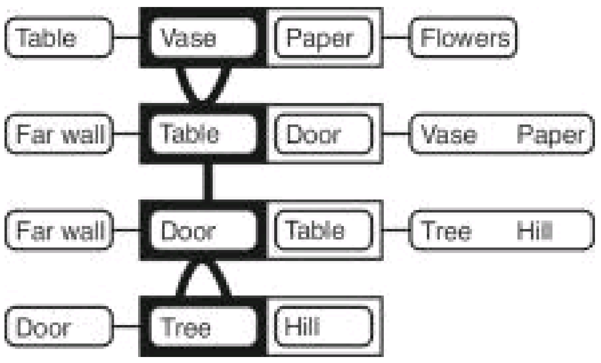
Whilst developing ICS (see Unit 7), May, Scott and Barnard (1995) developed a method of analysing user interfaces which takes into account human perception (a function of human cognition – see Unit 4). Consider the following scene. In this image we perceive a table in a room whose door is open. Outside we see a tree and a hill. Inside, on the table, we see a vase of flowers and a piece of paper. The key is that we perceive that this is what the collection of lines indicates – that the lines represent these objects. So, when looking at an image such as the one below, or a user interface, we apply psychological processes to interpret the image as a set of objects. Importantly, this perception of objects is focused. That is, we look at particular parts of an image and concentrate on them. For example, we might look at the table and then the objects that are on it (the vase with flowers and the paper), or we might look at the door and the objects we can see through it (the tree and the hill).



This view of perception assumes that our perception works hierarchically. The figure below illustrates a hierarchy for the objects in the figure above referred to as an object structure diagram. Using this hierarchical structure we can see that what we perceived is constrained by the level we are on.

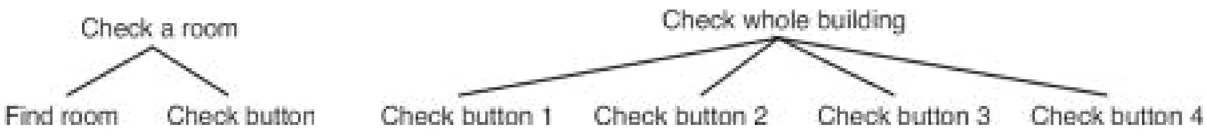


The changes in focus of perception as we view different parts of the scene are referred to as transitions. Below is a transition diagram of someone changing from focusing on the vase (containing flowers) on the table, to focusing on the tree outside. Transition diagrams show change in focus as a sequence of steps from top to bottom of a diagram. In the diagrams the object currently focused on is outline in black (in the first step below that is the vase). To the right of the focus object and within the same box are the focus object's siblings – those objects on the same level of the hierarchy. In the first step below the only sibling of the vase is the paper. To the left of the focus object is the parent of that object – the object that is directly above it in the hierarchy. For the case of the vase the parent is the table. Finally, to the left of the sibling objects are child objects of the focus object which in the case of the vase are the flowers. Thus, all possible changes in focus are thus shown on one line e.g. from the vase the focus could change to its parent (to the left in the diagram, the table in this case), a sibling (immediately to the right in the box, paper in this case), or its children (to the right of the box, flowers in this case). So there are three kinds of movement which need to be represented in the diagram. Movement up the hierarchy to the parent is indicated by È (the first transition below is up the hierarchy from the vase to the table – notice how the focus object changes in the next line, as do the parents, siblings, and children). Movement between siblings in the hierarchy is indicated by ½ as illustrated in the second transition below from table to door (both children of the far wall, hence the far wall stays as the parent between steps 2 and 3). Finally, there are movements from parent to children which are indicated by Ç as illustrated in the third transition from door down to its child tree.

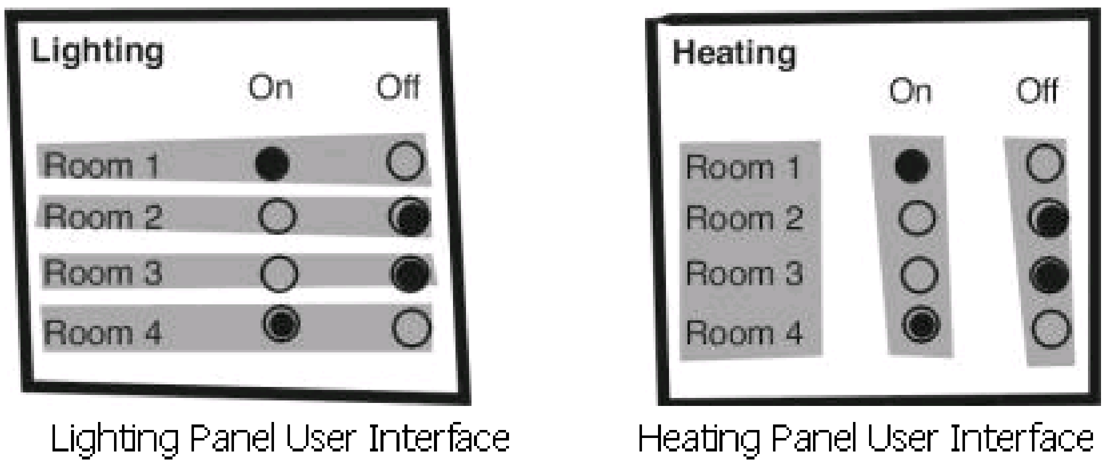


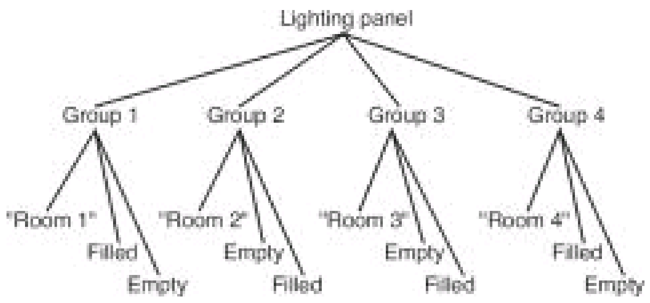
You might be wondering what on earth this has to do with evaluating user interfaces. Such understandings of perception come into play when we consider the tasks users are to perform and the user interfaces they use to complete their task. May et al. give an example of a caretaker of a building who, among their many other tasks, has two tasks which are to either check the heating and/ or lighting in a particular room, or to check the heating and/ or lighting in the whole building. Rather than actually having to go to each room to check it the caretaker has a display panel in their office which shows the status of lighting and heating in each room. The two tasks require different user interfaces to this display panel as discussed below – the key is that good design matches task structures to user interface structures.

Assuming that there are four rooms in the building, the following diagram illustrates the hierarchical composition of the two caretaker tasks we are considering (such diagrams are referred to as task structure diagrams – it is important not to confuse task structure diagrams and object structure diagrams).

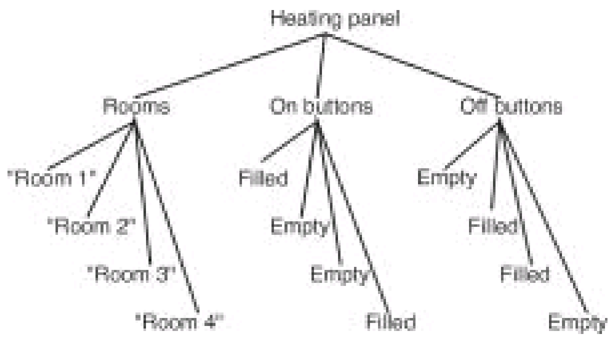


The caretaker has two interactive display panels in his office which are shown below. For each room there is an indication of whether the lighting or heating is on or off. The caretaker can press any of these buttons to change the state of the lighting or heating. These illustrations are followed by the object structure diagrams of these users interfaces – that is, how the objects in the display are perceptually grouped (by relative position of objects and shading in these interfaces). From these object structure diagrams and our task structure diagrams above we shall evaluate the interfaces with respect to the tasks.



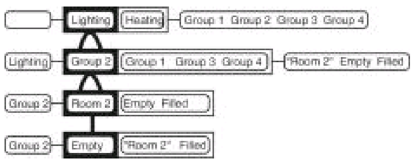


Lighting Panel Object Structure

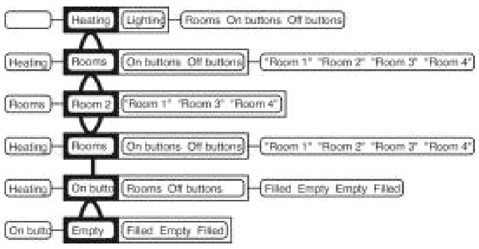


Heating Panel Object Structure

Now we have descriptions of the structure of the tasks and the object grouping in the interface. From these we can make assessments of the suitability of the different interfaces for different tasks. Consider the task of turning the heating and lighting on in room 2. When turning the lights on in room 2 the caretaker focuses on the lighting panel, then finds group 2, then clicks the empty button to turn it on. In contrast, to turn the heating on the caretaker first focuses on the heating panel, then the rooms group to find room 2, then changes to the on buttons group to select the appropriate button to turn the lights on. These two different ways are illustrated in the following transition diagrams. The extra transitions involved in using the heating panel tell us that the organisation of the lighting panel is better for the task of turning something on in a specific room. However, this may not be the most appropriate object grouping for other tasks.



Lighting transition diagram



Heating transition diagram

Activity 7 – Perception Based Evaluation

Construct the object structure for the remote control example used in the cognitive walkthrough section. Then consider the transition diagram for the given task of setting the video to record. What problems can you identify using this approach?

A discussion on this activity can be found at the end of the chapter.

Review Question 4

In the discussion of perception based evaluation an example involving interfaces to lighting and heating panels was given. Transition diagrams for turning the light and heating on in one room were given and the differences discussed for the different user interfaces. Construct the transition diagrams for checking whether the lights and heating are on in each room. What does that tell you about the suitability of the two user interfaces for this new task?

Answer at the end of the chapter.

GOMS

Perception based evaluation used an implicit notion of perception developed from psychological understandings of cognition. GOMS contrasts this by using an explicit model of cognitive processes developed from studies of users. GOMS itself stands for Goals, Operators, Methods, and Selection rules (see Unit 7). The quantitative predictions of execution time produced by GOMS can be used to compare designs. For example, to compare the time taken to delete all the files in a directory with a command-line and a graphical user interface, we would see the following predictions:

Command-line			
Step	Description	Op.	Time
1	Recall command DEL to delete files	Tm	1.35
2	Type DEL *¿	Tkx6	0.72
Total time (s)			2.07

GUI			
Step	Description	Op.	Time
1	Move pointer to one corner of directory window	Tp	1.10
2	Click (and hold)	Tb÷2	0.10
3	Drag selection to opposite corner	Tp	1.10
4	Let go of mouse button	Tb÷2	0.10
5	Press and hold mouse button to drag contents of window	Tb÷2	0.10
6	Drag to trash	Tp	1.10
7	Let go of mouse button	Tb÷2	0.10
Total time (s)			3.70

From this analysis a command-line interface would be quicker to use to delete the contents of a directory than a graphical user interface. Of course, this assumes that the user is an expert and knows how to issue the appropriate command to delete the files. The command-line interface is likely to be even better for deleting files using wildcards e.g. deleting all files ending in .txt.

Summary of GOMS

GOMS can be used to compare different user interface designs, and to support profiling of interface – describing the interface in terms of the time taken to perform various tasks. However, the evaluator must pick the users' tasks and goals which may mean that evaluators only find what they are looking for.

The application of GOMS is not as easy as heuristic analysis, or cognitive walkthrough (discussed previously) as it takes lots of time, skill, and effort to complete an analysis. Also, as it is concerned with the time it takes to complete tasks, GOMS does not address several important UI issues, such as readability of text, memorability of icons and commands, and does not address the social or organisational impact of introducing new systems into the workplace.

Activity 8 - GOMS

GOMS is intended to predict time to complete tasks for expert users. What tasks are there that can be completed quicker by and expert using a mouse than a keyboard? Use GOMS to compare keyboard and mouse interactions in terms of predicted task execution time.

A discussion on this activity can be found at the end of the chapter.

Summary of Model based Evaluation

In this section we have discussed model based evaluation which differs from the previous forms of evaluation in that nobody actually attempts to use the system or design in order for it to be evaluated. The approaches covered can give several qualitative and quantitative predictive measures of user performance with the system. The key is that they are predictive – they attempt to give some idea of what a use probably would do with the system. Furthermore, the models can be used to attempt to explain why the results are what they are. For example, the results of perception based evaluation can be explained in terms of understandings human perception. Even though constructing the models and running the evaluation in this way can be time consuming, it is still less work that user experimentation. Moreover, the models developed for such evaluations can easily be modified when the design changes. Therefore we can quickly get some idea of the impact of changes in the design. An important future development will be the construction of tools to assist evaluators in using model based approaches as they can often be difficult to learn and understand.

Review Question 5

Discuss the difference between approaches to evaluation which use models, and those that do not. Consider their appropriateness and the kinds of evaluation they can be used for.

Answer at the end of the chapter.

Summary of Evaluation

In this unit we have looked at several different evaluation techniques. These were classified according to who were participants in the evaluation – either potential users, trained evaluators, or models of cognition. There are many more differences between the evaluation techniques which have been alluded to throughout this unit. This section will draw out these differences so that you get a better idea of how to use different techniques, and when they are appropriate. Preece (1995) identified several differences between evaluation techniques which are discussed in the remainder of this section.

Purpose of the Evaluation

As mentioned in the introduction, Preece identified four main purposes for doing evaluation which are outlined below. Clearly the purpose of the evaluation dictates the kind of evaluations that we should perform and the data we would want to collect. These issues are also outlined below:

- Engineering towards a target – asking whether the system we have designed is good enough yet. The question we have to ask ourselves here is what targets are being engineered towards. If it is in terms of user satisfaction, for example, then we might employ a technique such as user studies where we can get qualitative information about whether the user is satisfied with the system.

- Comparing designs – to identify which designs are best for a given set of tasks. In general, most evaluation approaches can give us some comparative evaluation, but quantitative results are usually the easiest to use for comparisons e.g. the results produced by user experimentation, some user studies, and GOMS.

- Understanding the real world – working out how well the design would work in the real world such as an office or other workplace. Approaches such as GOMS and user experimentation do not give us information about how the system will fit into the real world. Other approaches such as user studies (where the system could actually be evaluated in the real world site) would be more appropriate for such questions.

- Checking conformance to a standard – whether the system meets standards that have been set. Again we need to consider what the standards are that we are trying to conform to. If they are that a task should be performed within a specific time limit then we could use an approach such as GOMS to predict the task completion time.

Stage of System Development

As we discussed at the start of this unit, performing evaluation throughout a system's development provides us with a much more flexible process in which potentially time consuming problems are identified throughout development. The stage at which evaluation is performed will determine which kinds of evaluation techniques are appropriate. At the very beginning of the process interview and questionnaires are probably the most appropriate as they can give us plenty of information to inform the design. Once some more rigorous designs have been developed we might try using GOMS or ICS to compare designs for specific issues such as task completion time or difficulty to learn. If we have more time and effort, more grounded results (i.e. not just predictions) could be obtained from user studies, heuristic evaluation, cognitive walkthrough, and user observation. In some cases involving users very early in the design process may be problematic as they may be put off by the incomplete nature of designs at that stage.

Type of Data

As discussed throughout this unit there are two main kinds of data which can be collected from evaluations: quantitative (numerical values such as time, or attitude ratings) and qualitative (such as opinions). These two types of data provides answers to different kinds of questions, and moreover, are generated by different kinds of evaluation. The key is to select the evaluation technique which produces the kind of data necessary to answer the questions being asked. The table below lists the kinds of data we might expect to be able to get from the different kinds of evaluation we have discussed in this unit.

Technique	**Quantitative data**	**Qualitative data**
User observation	X	X
Interviews		X
Questionnaires	X	
User experimentation	X	
Cognitive walkthrough		X
Heuristic evaluation		X
Perception based evaluation	X	
GOMS	X	
ICS	X	

Review Question 6

In the summary of this unit we discussed four different purposes of evaluation. Complete the table below to indicate which techniques are appropriate or not for the purposes given (mark a tick or cross).

Technique	**Engineering towards a target**	**Comparing designs**	**Understanding the real world**	**Checking conformance to standard**
User observation				
User feedback				
User feedback				
Cognitive walkthrough				
Heuristic evaluation				
Perception based evaluation				

Technique	**Engineering towards a target**	**Comparing designs**	**Understanding the real world**	**Checking conformance to standard**
GOMS				
ICS				

Answer at the end of the chapter.

Considerations

Different kinds of evaluation require different time, effort, number of people involved, and equipment. It is important to consider whether a certain kind of techniques is appropriate for the stage of development.

We then need to consider how valid the data we have collected is. This refers to whether the data collected is suitable for the purpose of the experiment. If we were to try to use GOMS to predict the time it would take a novice user to complete tasks using a system we would get invalid results as GOMS is designed to predict task completion times for expert users, not novices.

Our data might be valid, but then we have to ask ourselves whether it is reliable. That is, whether we can expect the same results consistently. Clearly, for model based evaluation this depends on the way in which we use the models. Similarly, when performing evaluator based evaluation such as heuristic evaluation, it depends on how well the evaluators are trained, and whether we have enough to produce reliable results. When users are involved, well designed experiments should give reliable experiments, whereas user observation tend to give quite unreliable results. We might be concerned with reliability when considering whether the system meets certain standards – meeting standards needs to be shown by reliable evaluations.

Finally, we need to be aware of the biases that we may be introducing into the results of our evaluations. Preece identified two main sources of bias:

- **Selective data gathering** – concentrating on certain aspects of the situation and not taking into consideration others which may also be important.
- **Manipulation of the evaluation situation** – for instance, asking leading questions in interviews rather than letting interviewees formulate their own answers.

Discussion Topics

There are many new concepts in this unit. If you want to discuss with your colleagues or make comments about the concepts to them, use the on-line facilities.

- Why should we use evaluation?
- What techniques are appropriate at different stages of system development?
- If heuristic evaluation can identify usability problems, why bother using other approaches?

Answers and Discussions

Answer to Review Question 1

Concurrent protocols involve the subjects of the observation verbalising what they are doing and why whilst they perform the task that is the subject of the observation. Retrospective protocol is used after the observation – subjects recount what they did and why.

Concurrent protocols may distract the subject whilst they are attempting to perform their task. On the other hand, retrospective protocols rely on the subject remembering what they did and why, and involve the subject in extra time as part of the evaluation.

Answer to Review Question 2

There is only one independent variable which is the type of user interfaces. It has three levels: 1) command-line, 2) desktop metaphor, and 3) virtual environment.

There is one dependent variable – the time it takes to complete the take using a given interface.

As there is only one independent variable and it has three levels there are three conditions – the three levels of the independent variable.

Confounding variables might include: subjects' experience of any of the interfaces, and typing experience (for command-line interface).

Answer to Review Question 3

Heuristic 1.1: The interaction is concise, but not too clear about how to proceed from step to step

Heuristic 1.2: The language is not natural – users have to press PGRM to start the video programming process, also, OK does not do what users might expect (accept a value).

Heuristic 1.3: Users must remember both what PGRM does, and that the first set of numbers presented are not the channel numbers.

Heuristic 1.4: Consistency seems good – the user always presses Þ to move to the next entry.

Heuristic 1.5: Feedback is good – something always flashes when there is something to be done.

Heuristic 1.6: There are no clearly marked exits. In fact the user must press PGRM again to exit.

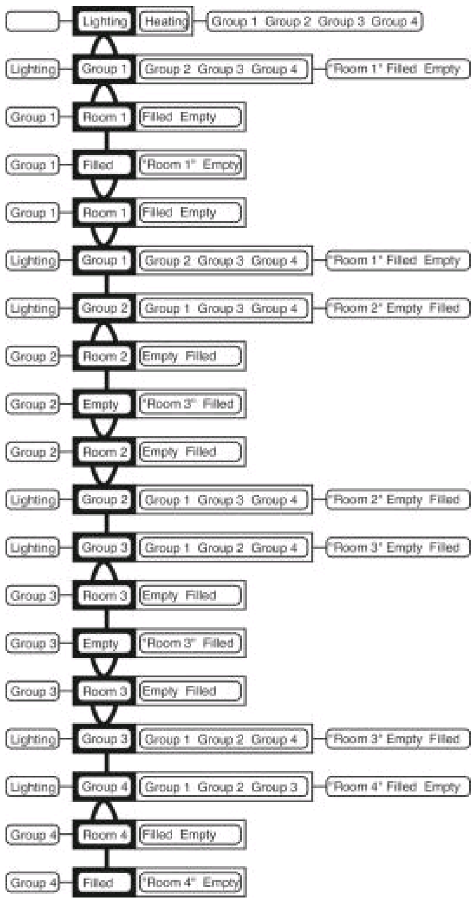
Heuristic 1.7: Shortcuts are not supported – its not clear how relevant they would be to a remote control anyway.

Heuristic 1.8: There are no error messages.

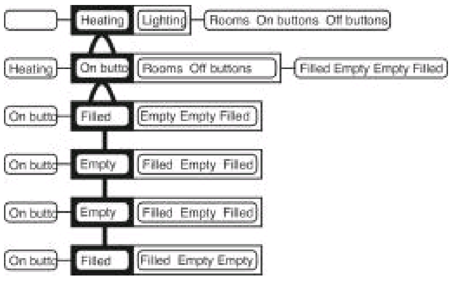
Heuristic 1.9: It is not possible to enter erroneous information (this is not apparent from the example).

Heuristic 1.10: There is no help or documentation provided on the remote control.

Answer to Review Question 4


Lighting Heating Group 1 Group 2 Group 3 Group 4
Lighting Group 1 Group 2 Group 3 Group 4 "Room 1" Filled Empty
Group 1 Room 1 Filled Empty
Group 1 Filled "Room 1" Empty
Group 1 Room 1 Filled Empty
Lighting Group 1 Group 2 Group 3 Group 4 "Room 1" Filled Empty
Lighting Group 2 Group 1 Group 3 Group 4 "Room 2" Empty Filled
Group 2 Room 2 Empty Filled
Group 2 Empty "Room 3" Filled
Group 2 Room 2 Empty Filled
Lighting Group 2 Group 1 Group 3 Group 4 "Room 2" Empty Filled
Lighting Group 3 Group 1 Group 2 Group 4 "Room 3" Empty Filled
Group 3 Room 3 Empty Filled
Group 3 Empty "Room 3" Filled
Group 3 Room 3 Empty Filled
Lighting Group 3 Group 1 Group 2 Group 4 "Room 3" Empty Filled
Lighting Group 4 Group 1 Group 2 Group 3 "Room 4" Filled Empty
Group 4 Room 4 Filled Empty
Group 4 Filled "Room 4" Empty

Checking lighting in every room


Heating Lighting Rooms On buttons Off buttons
Heating On butto Rooms Off buttons Filled Empty Empty Filled
On butto Filled Empty Empty Filled
On butto Empty Filled Empty Filled
On butto Empty Filled Empty Filled
On butto Filled Filled Empty Empty

Checking heating in every room

This shows that the heating panel is better for checking all the rooms than the lighting panel as the ‘on' buttons are all in the same group.

Answer to Review Question 5

Model based approaches use simple models of human cognitive processes to predict how users would react to systems. As such they do not require users, but are often complex to use. Other approaches rely on observing, or questioning users about their use of a system. This means that users have to be found who are willing to try to use a system. Moreover, their actions need to be interpreted by analysts, and sufficient users need to be found to give generalisable results. Finally, some approaches do not rely on users, but use trained analysts. These benefit from not requiring users, but results may be skewed by analysts prior understandings of what the evaluation is about – what sort of problems they expect to find.

Approaches which involve models are appropriate where we want to make quite specific predictions about user behaviour, possibly before any systems have been built. These should help us to decide whether certain interfaces would be more appropriate than others for given tasks.

Answer to Review Question 6

Technique	**Engineering towards a target**	**Comparing designs**	**Understanding the real world**	**Checking conformance to standard**
User observation	Yes	Yes	Yes	No

Technique	**Engineering towards a target**	**Comparing designs**	**Understanding the real world**	**Checking conformance to standard**
User feedback	Yes	Yes	Yes	No
User feedback	No	Yes	No	No
Cognitive walkthrough	No	Yes	No	Yes
Heuristic evaluation	Yes	Yes	No	Yes
Perception based evaluation	Yes	Yes	No	No
GOMS	Yes	Yes	No	No
ICS	Yes	Yes	No	No

Discussion on Activity 1

Four advantages of the waterfall model:

- It provides a comprehensive template in which many important aspects of design and development can be placed.
- The steps of the model are in some way generic steps that are found in most software-engineering paradigms.
- It is claimed to be the most widely used model of software design and development, at least among large software projects – this means that there is a large body of people who would understand how a project would develop under such a model.
- It is arguably better than a haphazard approach to design and development, at least from the point of view of being able to manage the production of software.

Discussion on Activity 2

Fourth generation tools allow systems to be specified at a high level of abstraction and then convert this specification directly into runnable code. It is similar in structure to the iterative prototyping discussed previously. However, instead of prototyping, complete systems are generated from the requirements. These can then be evaluated to determine whether they are suitable. If not, the specification (derived from consultation with clients) will need to be changed, or the style of system generation altered i.e. changing the kinds of interfaces the code generator tool produces.

Discussion on Activity 3

Out experiment would have to take into account the kind of user, the working environment, and the kinds of tasks and interaction users have with computers. For example, a user who relies heavily on email correspondence, works in their own office, and does not perform intensive interaction with their computer may prefer a dialogue box with the sound of a trumpet. In contrast, someone who does not rely on email and moreover, performs intensive interaction with their computer e.g. an air traffic controller, may want no visual notification, and only slight audio indication such as a coughing noise. Therefore, our hypothesis might be that there is a correlation between the users' tasks, email reliance, and notification style.

Independent variables:

- Audio alert: none, beep, quack, cough, trumpet
- Visual alert: none, flashing icon, dialogue box

- Reliance on email: none, medium, high
- Reliance on email: none, medium, high

Dependent variables:

- User preference

A large problem will be finding enough subjects to complete the experiment – there are four independent variables, each with several levels. Another problem would be determining how intensive a task is in order to assign the subjects appropriately – maybe this could be determined by some pre-experiment tests. Finally, there is the problem of measuring user preference (the dependent variable) – this could be ascertained from post-experiment interviews or questionnaires.

Using a questionnaire we could ask people what kinds of noises get their attention, what noises they would consider suitable for a working, home, or recreational environment. Maybe this could be structured using multi-choice answers, or possibly open questions in order to find a greater range of sounds. A harder aspect of the questionnaire would be how to find out which visual indicators people might prefer. In order to cater for non-computer users we would have to think of some metaphors for email arrival. For instance, likening it to post arriving every five minutes – would they was some one coming up to them and announcing it (like a dialogue box), or would a simple indicator such as a lamp lighting be better (like a flashing icon).

Understanding non-computer users' preferences can help in the design of new systems – such understandings can help us to make new systems more friendly to potential new users.

Discussion on Activity 4

Experimentation typically requires a large number of subjects who are given quite restricted tasks to perform. User observation, on the other hand, requires a smaller number of subjects who may carry out larger and more realistic tasks. Moreover, whereas experimentation typically requires quite specific physical surroundings such as a usability lab, user observation is more flexible and can even be employed in the workplace. Results from the two approaches also differ - experiments provide quantitative data, whereas user studies typically give qualitative data.

Discussion on Activity 5

The table below details possible problems with the remote control in terms of the **first** ten heuristics

Heuristic	**Problem**	**Severity**
1.1	The interaction is concise, but not overly clear	1
1.2	There is plenty of jargon in the interface which can be confusing e.g. PGRM	3
1.3	The memory load is relatively low as each task is constrained and simple. However, they must remember the sequence in which items must be entered, and the concept of numbered video recordings timers	3
1.4	The interface is consistent	0
1.5	Feedback on the current input point is provided – typically by flashing the appropriate digits	0

Heuristic	Problem	Severity
1.6	Feedback on the current input point is provided – typically by flashing the appropriate digits	4
1.7	There are not shortcuts, but as the system is minimal anyway, the utility of these seems dubious	0
1.8	There are no error messages, but incorrect information is difficult to enter	0
1.9	Again, it is difficult, if not impossible, to enter incorrect information	0
1.10	Again, it is difficult, if not impossible, to enter incorrect information	2

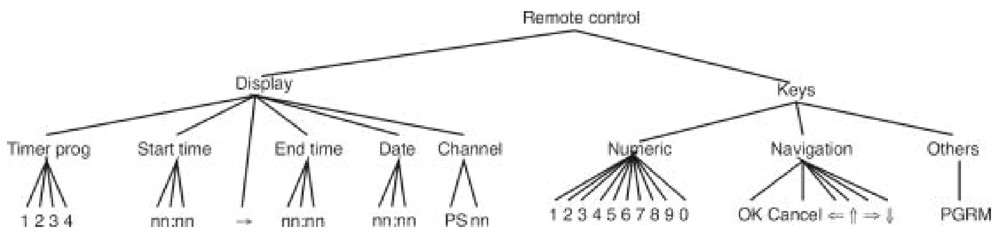
The main interface improvements would be some clearly marked exits, less jargon, and the possibility of flexible data entry. It would also be beneficial to have on-line help which may be audio in the case of such a small device.

Discussion on Activity 6

The third and fourth steps are much more coherent than the first two. For a start, the number blinks to indicate that it can be changed (question 2, 3). Also, the user can enter the number directly on the keypad, e.g. 22, or can use the arrows to increase or decrease it. The numbers themselves immediately update, so meeting question 4. As with the previous steps, it is a bit odd that the user has to press Þ instead of OK, but by this point the user may have become accustomed to this interaction.

In terms of redesign, the main changes would be to allow OK to be used to confirm entries. Also, it would be clearer if the video timers were referred to by letters rather than number which may be confused with channel numbers – maybe these could even be entered later in the task so that the desired channel is set first.

Discussion on Activity 7



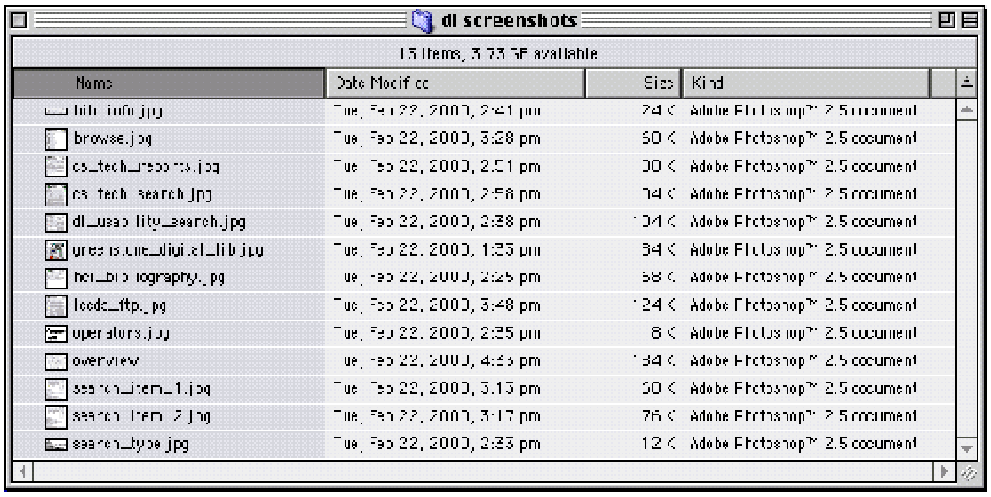
In setting the video two approaches can be taken in entering numeric values such as time and date – using the numeric keys, or the navigation keys. Both of these approaches indicate possible problems due to the structure of the interface. Using the navigation keys, the OK and Cancel buttons are in the same group, so implying that they might be used to confirm entry of values – this is not the case. Conversely, entering numeric values using the numeric keys means that moving to a different entry point (e.g. from start time to end time) requires shifting focus to the navigation group to press the Þ key which requires effort on the part of the user.

Discussion on Activity 8

Even though we assume expert users when evaluating systems using GOMS there are still plenty of tasks for which a mouse is more efficient than a keyboard.

For example, supposing a user wants to delete the files bib_info.jpg, cs_search.jpg, and greenstone_digital_lib.jpg from the file directory illustrated below. One way of doing this using a mouse driven interface would be to select bib_info.jpg, then drag it to the trash, then select cs_search.jpg, and drag it to the trash, and finally select greenstone_digital_lib.jpg and drag it to the trash. This is 3(Tp + Tb÷2 + Tp + Tb÷2) = 7.2s. To perform the same task using a command-line interface such as DOS would require the user issuing a series of commands such as ERASE bib_info.jpg¿, ERASE cs_search.jpg¿, ERASE greenstone_digital_lib.jpg¿. This would take 19Tk + 20Tk + 34Tk = 8.76s. Therefore, in this case, it would be quicker for an expert to use a mouse to delete these files.

Clearly an expert DOS user could use wildcards to shorten the filename and so reduce the number of keys pressed e.g. ERASE g*¿ would erase greenstone_digital_lib.jpg. However, we could argue that this would involve a mental operator (Tm). Moreover, the mouse based approach could be made more efficient by multiply selecting the files and then dragging them all at once to the trash so saving click and drag operators. The key to this example is that mouse based interfaces are better for multiple file operations where wild-cards would be difficult to use.


dl screenshots

13 items, 3.73 GB available

Name	Date Modified	Size	Kind
bib_info.jpg	Tue, Feb 22, 2003, 2:41 pm	24 K	Adobe Photoshop™ 2.5 document
browse.jpg	Tue, Feb 22, 2003, 3:28 pm	60 K	Adobe Photoshop™ 2.5 document
cs_tech_reports.jpg	Tue, Feb 22, 2003, 2:51 pm	30 K	Adobe Photoshop™ 2.5 document
cs_tech_search.jpg	Tue, Feb 22, 2003, 2:58 pm	74 K	Adobe Photoshop™ 2.5 document
dl_usability_search.jpg	Tue, Feb 22, 2003, 2:38 pm	104 K	Adobe Photoshop™ 2.5 document
greenstone_digital_lib.jpg	Tue, Feb 22, 2003, 1:35 pm	34 K	Adobe Photoshop™ 2.5 document
hci_bibliography.jpg	Tue, Feb 22, 2003, 2:25 pm	58 K	Adobe Photoshop™ 2.5 document
leeds_ftp.jpg	Tue, Feb 22, 2003, 3:48 pm	124 K	Adobe Photoshop™ 2.5 document
operators.jpg	Tue, Feb 22, 2003, 2:35 pm	8 K	Adobe Photoshop™ 2.5 document
overview	Tue, Feb 22, 2003, 4:55 pm	134 K	Adobe Photoshop™ 2.5 document
search_item_1.jpg	Tue, Feb 22, 2003, 3:13 pm	50 K	Adobe Photoshop™ 2.5 document
search_item_2.jpg	Tue, Feb 22, 2003, 3:17 pm	76 K	Adobe Photoshop™ 2.5 document
search_type.jpg	Tue, Feb 22, 2003, 2:33 pm	12 K	Adobe Photoshop™ 2.5 document

Chapter 10. Advanced Topic: CSCW

Table of Contents

Context

The purpose of this unit is to explore the important, evolving area of Computer Support of Collaborative Work (CSCW). Most work involves people working together. Increasingly, interactive computer systems are being used to support such shared activity. This unit discusses a wide range of system and considers how they may enhance business practice. Just as 'traditional' interactive systems design needs to be informed by cognitive psychology, developers of CSCW systems need to be aware of the potential social and work-practice impacts when these systems are introduced. You will consider some of these users-centred issues.

Objectives

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

- Explain the meaning of CSCW;
- Classify CSCW systems in terms of the broad ways they are used (e.g., remote vs. local use and synchronous vs. asynchronous use).;
- Describe a range of CSCW tools and explain their significance to business.;
- Discuss the role of sociology and group-working research to CSCW;
- Discuss several social and work-practice impacts of CSCW and suggest ways of solving potential problems.

Unit Outline

In this unit we are concerned with computer supported collaborative work (CSCW)– broadly a set of concerns related to computer support for activities in which more than one person is involved. Within previous units you have focused on the design process (units 2,3 & 6), the user (units 4,5 & 7) and defining the human computer interaction (units 8 & 9). Within CSCW there is the added complexity of the human-to-human interaction, which is important, as well as how technically mediating the process affects the interaction. Specifically this unit will review the background to CSCW as a separate yet related field within human computer interaction (HCI). To understand CSCW applications and design it is important to be able to define the different ways these systems are used and how this helps categorise the array of CSCW specific tools and applications. Finally the social impacts of cooperative work on CSCW and how the technology effects social interactions are established.

Introduction to CSCW

First of all we need to understand why CSCW has become such a crucial field of study within interactive systems.

With changes in business organisations and world economics there has been a move towards global markets that are multinational and multilingual organisations. Rising organisational costs have also driven increases in the degree of strategic co-operations and opportunistic alliances.

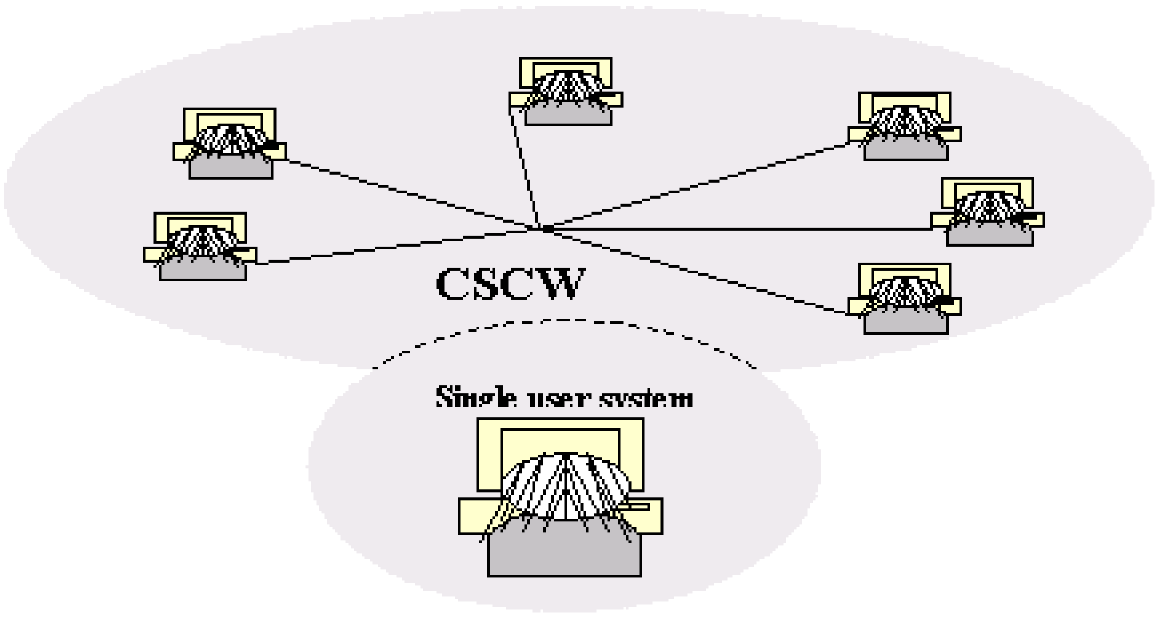
Both of these factors have produced an increased need for businesses to communicate and collaborate both locally and remotely.

Over the last two decades computer technology has evolved a worldwide infrastructure.

There has been an international increase in networking capabilities producing more potential for applications and tools characterised by the information superhighway. A spread of connectivity has created what has been termed the global village. Finally a merging of technologies has occurred between telephony, television and computing.

All of these factors have led to the increased potential for computing technology to aid collaborative working.

However, despite widespread accessibility of computer networking technologies and the reliance of many organisations on successful cooperative activities the historical focus within HCI (see unit 1) has limited the support provided by automated tools for the creation of single user independent systems. Since many people work in groups on cooperative tasks CSCW has sought to address these needs.



It is important to understand how CSCW systems differ from other interactive systems. Designers must understand the nature of cooperative tasks in order to design appropriate technology for the cooperative work setting. A set of questions must, therefore, be answered if appropriate CSCW systems and software are to be designed:

How do we define group work? How can computer systems and software that support group working be developed? What are the impacts of technology on group working?

To answer these questions CSCW involves inter-disciplinary researchers including psychology, sociology, organisational theory and anthropology.

A note on terminology

It is important to understand that CSCW is a wide-ranging term which encompasses not only the technology / software but its users and their physical and social environment. Groupware, however, is a term often referred to within CSCW as related to the technology and software used for computer supported group-working.

Review Question 1

Explain why CSCW systems have grown in their importance over recent years?

Answer at the end of the chapter.

Review Question 2

Why has the development of applications to support cooperative working been slower than other areas of computer support?

Answer at the end of the chapter.

Review Question 3

What are the three specific aspects of CSCW systems that must be understood to design appropriate applications and tools

Answer at the end of the chapter.

How CSCW is used

When designing computer supported collaborative work systems and applications it is important to understand how they will be used and for what types of tasks. However there has been much controversy over this issue, as conceptualisation of the field should not artificially or inadvertently preclude specific types of cooperative work. Characterising how these systems are used should therefore allow for the rich diversity of forms of cooperative work.

Who uses CSCW systems

It is important to note that CSCW systems have a varied set of collaborating users that will impact on what systems are required and how the systems are used. Collaborating users may have different:

- Goals
- Equality of status,
- Feelings of comradeship
- Organisational alliances
- Heuristics and conceptual frameworks

It is important to understand that whilst some groups do have a closed and predetermined character, such as project teams, others are formed and disbanded spontaneously when the need arises. This distinction has been termed the difference between formal and informal groups (Preece, 1995). CSCW systems are increasingly being developed to support informal group collaborations as they are recognised as an important characteristic of the work scenario.

The geographical location of the user

The geographical distribution of group members is an important aspect of group interaction. Cooperative systems are often classified as either local, with group members co-located in the same environment or remote with members at different locations. This divide is concerned as much with the accessibility of users to each other than their physical proximity. Computer support for these interactions has traditionally considered the case of remote asynchronous (at different times) group working. However, more recent research has aimed to support synchronous (at the same time) face-to-face meetings.

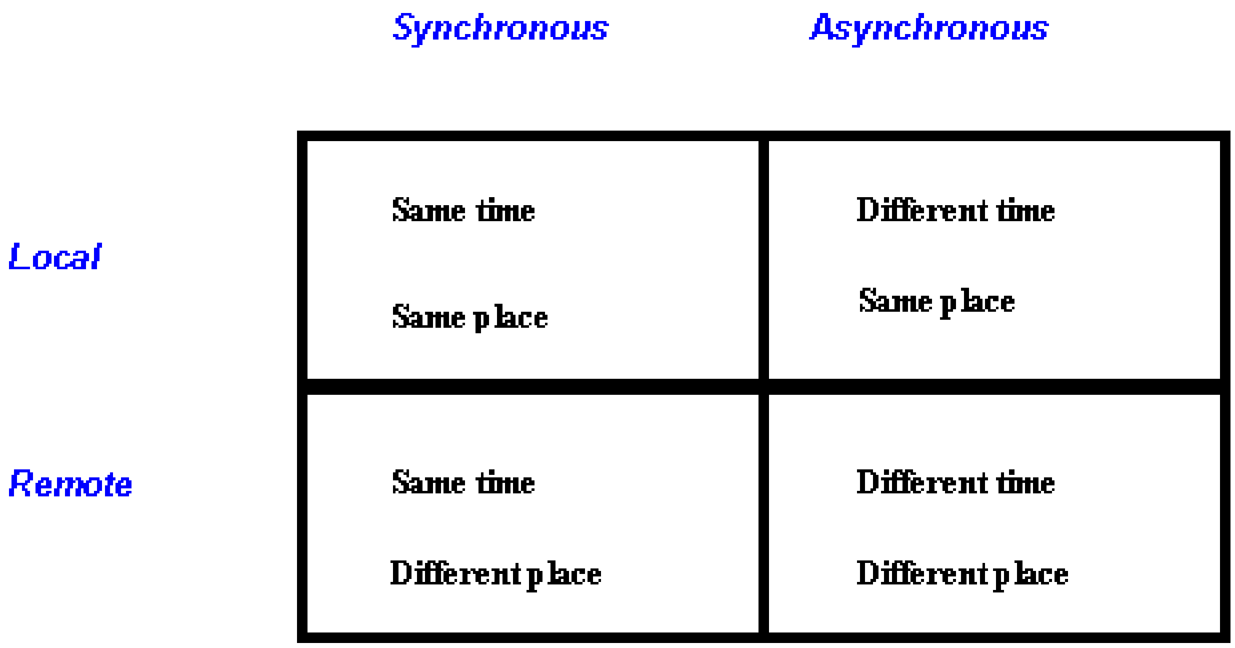
A note on terminology

The term ‘co-located’ is often referred to instead of ‘local’ to emphasise this logical division. The term local used in this context should not be confusion with the distinction between remote and local communication systems.

The form of interaction

The most commonly used classification for the mode of interaction is between asynchronous (occurring at different times) and synchronous (occurring at the same time) working (Preece et al, 1995; Dix, 1998) . These two forms of interaction are important in distinguishing computer support for different tasks. Synchronous cooperation is ideal for creative problem solving, as it often requires immediate input from each group member (i.e. to bounce ideas off of). In contrast, rigid tasks often have a previously formulated strategy whereby group members take on a particular role and work in an asynchronous manner.

Interactions between the form of interaction (synchronous, asynchronous) and the location of the user (local, remote) are often used to define different modes of interaction for CSCW systems. This approach is probably often used because it easily relates to CSCW applications and tools. However as the types of interactions supported by CSCW systems become ever more complex this matrix type approach becomes inadequate.



	Synchronous	Asynchronous
Local	Same time Same place	Different time Same place
Remote	Same time Different place	Different time Different place

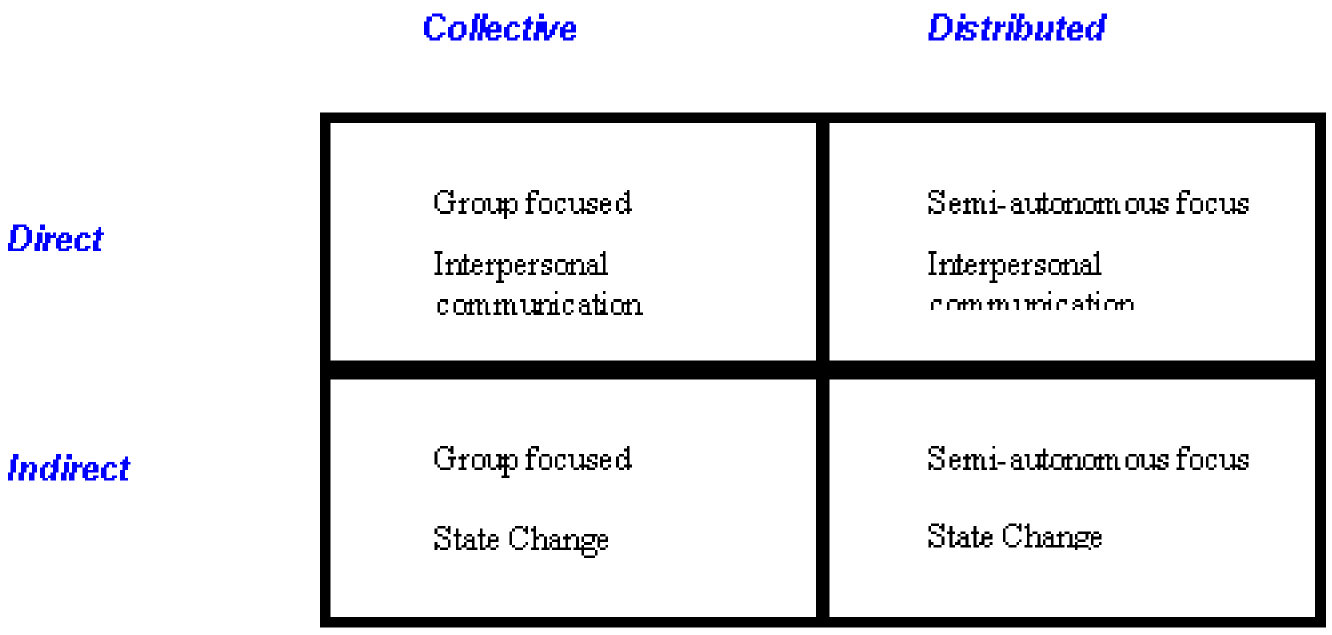
Group working has been further categorised in a number of different ways. The group work and thus the decision making process can be conducted:

- collectively, whereby the group has focused strategies
- distributed across the group whereby decisions are made by an ensemble of semi-autonomous workers planning and acting on their own strategies.

Furthermore, cooperative working may be conducted:

indirectly, mediated and guided by the changing state of the process; or directly mediated and guided by direct interpersonal communication.

Interactions between these 4 dimensions can be used to analyse collaborative tasks in more depth. These positions indicate what type of support is required from the technology and potential problems.



	Collective	Distributed
Direct	Group focused Interpersonal communication	Semi-autonomous focus Interpersonal communication
Indirect	Group focused State Change	Semi-autonomous focus State Change

Review Question 4

How would you define a group that has a scheduled, pre-set agenda?

Answer at the end of the chapter.

Review Question 5

Consider the task of 'writing a co-authored book' and the task of 'brain-storming an advertisement campaign', which task would require synchronous or asynchronous cooperation to be completed most efficiently.

Answer at the end of the chapter.

Review Question 6

A group focused collaborative exercise which is guided by changes in the task can be defined as a direct collective collaboration (True / False)

Answer at the end of the chapter.

CSCW tools and applications

The number and variety of CSCW tools and applications (often known as groupware) is quickly growing as are the number of ways to classify them. The following three categories are a simplification of various authors' groupings (Baecker, 1993; Preece, 1995; Dix, 1998) :-

Computer mediated communication:

- Email and bulletin boards, structured message systems
- Videoconferences / desktop video conferencing / multicasting
- Virtual collaborative environments

Meeting support:

- Augmentation tools

- Meeting rooms

Decision support systems and tools

- Shared work surfaces, PC’s and window systems
- Shared editors and co-authoring systems
- Shared diaries, active badges

Computer mediated communication

Computer mediated communication is the main research focus and the most used form of CSCW (Baecker, 1993; Keisler, 1997). Email and bulletin boards are the simplest and yet most successful asynchronous CSCW systems (although there are synchronous variations). The growth of distributed organisations has been attributed, in part, to email as an effective form of computer mediated communication. This communication medium has led to the emergence of on-line communities with their own specific HCI and social interaction problems e.g. information overload, lack of recipient feedback, inappropriate informality etc.

The history of videoconferencing has developed in parallel with advances in the technology (Baecker, 1993). Listing them in chronological order multimedia communication started first with close-caption TV (CCTV) with dedicated lines transmitting (synchronous) video directly to participants. Video conferencing began with the transmission of group images from one room to another via a common monitor. Multimedia communication really came into its own with the advent of desktop videoconferencing. Users sit in front of their computer and communicate in real time (synchronously) via a microphone, camera and (often) a digital workspace. This configuration is often referred to as a picture-in-a-picture (PIP) setup or CuSeeMe. The communication can take place on a point-to point basis or can involve many individuals and sites.

With the introduction of media spaces, distributed users access one another, like videoconferencing systems, via video and audio links. However, media spaces not only support explicit, intentional interactions and shared artefacts but also informal interactions and awareness. These informal communications (colleague presence, activity and availability awareness, and unplanned interactions) have been identified as critical to effective group work. Awareness technologies have evolved to allow distributed workers awareness of their co-workers and of their potential for collaboration. These technologies invariably use video images as the main data source for awareness although there are audio only awareness tools.

Internet-based videoconferencing has been regularly used since the early 1990’s . However, the first major enhancement to Internet videoconferencing has been the advent of multicasting. A Unicast connection transmits data on a point-to-point basis whilst a Multicast connection allows for data to be transmitted to multiple recipients. During multicasting the network replicates (at the routers) the packets transmitted. The replicated packets can be sent to as many recipients as have requested the data and are members of the multicast group. In multicast conferencing, audio and video are sent in different streams. Desktop conferencing facilities on computer workstations, can use a combination of multicast conferencing tools.

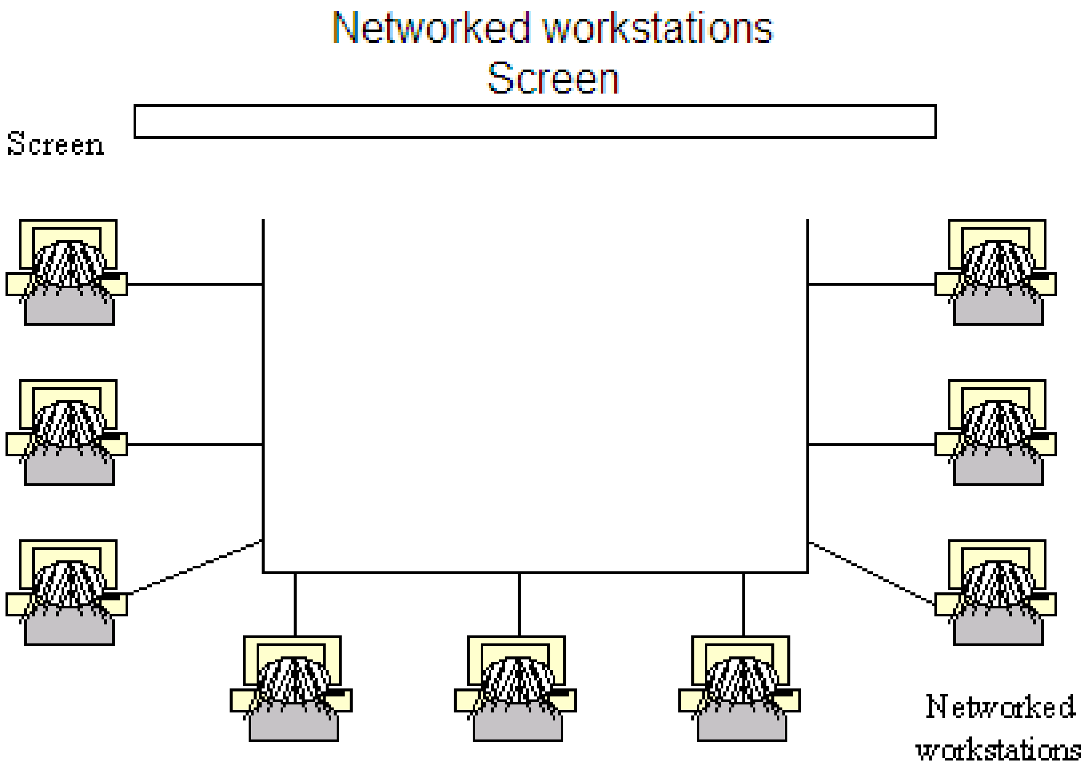
Virtual reality is a computer-based application which allows human-computer and human-human interactivity through a sensory environment called the virtual world which is dynamically controlled by the user’s actions. Virtual environments rely heavily on the notion of immersion both physically and cognitively. Keyboard and monitor input devices allow a user to be partially immersed whilst head mounted displays produce total-immersion in the environment. A user is cognitively immersed in the environment when they feel immersed in the action.

Collaborative virtual environments provide remotely located users with the ability to collaborate via real interactions in a shared artificial environment. Virtual reality communication environments have been argued to provide a natural, intuitive environment for communication with the added benefit of removing some of the social taboos from social interactions. Virtual reality animated actors called avatars aid the user in their interaction with others in the virtual reality environment making it seem more natural in two ways. Firstly, avatars can represent the user within the environment - a user relates and collaborates with other users via their avatars. Secondly, an avatar can represent a software agent with the actor's behaviour defined by that agent.

Meeting support systems

Meeting support applications and tools have emerged as a form of support for local synchronous meetings rather than with computer mediated communication, which has traditionally been primarily a support for remote synchronous & asynchronous collaborations (Preece, 1995; Keisler, 1997; Dix, 1998).

Meeting room systems often take the form of a meeting room furnished with a large screen video projector and a number of computer workstation / terminals.



Most of these systems focus on improving decision-making by groups rather than individuals. The large screen and all the participants' screens show the same image and this tool is often used as an electronic whiteboard easily accessible by all the group members.

Argumentation tools support and document multiparty arguments and negotiations. Many argumentation tools have been developed to support system designers in the decision making process. Tools that support argumentation often have a hypertext type format which allows designers to work simultaneously (synchronously) although they are often used asynchronously by local group members.

Co-authoring and shared applications

The aim of co-authoring systems is to support the collaboration necessary between co-authors in producing a document. The general mode of interaction for these systems is asynchronous cooperation with each user working as a semi-autonomous individual on a portion of the document. Many of these systems use a hypertext model. The text forms the basis of the interaction with linked networks of data (usually text or graphics) connected to this basic structure. Shared editors, in contrast to co-authoring systems, work synchronously with text or graphics. Specific problems occur with this synchronous mode of interaction.

Shared diaries and calendars are often used for CSCW purposes. This idea has been taken further with active badges which track the whereabouts of users within a building and then relay the information back to a shared diary system. The accessibility of the resultant information has many social impacts that are becoming more evident especially with reference to privacy issues (see the next section)

Shared workspaces and window systems are examples of synchronous collaboration applications.

An important aspect of these shared applications is the notion of floor control – who has control over the application at any one time. There is either a social protocol in force or technical mechanisms imposed by the system which work on a first-come / first-served or turn-taking basis.

Activity 1 – Classifying applications 1

Identify some different tools & applications that would support each of the four classifications of group cooperation described according to geographic location and mode of cooperation i.e. remote synchronous, remote asynchronous, local synchronous, local asynchronous

A discussion on this activity can be found at the end of the chapter.

Activity 2 - Classifying applications 2

Can you think of different tools & applications that would support each of the four classifications for mode of group cooperation i.e. Direct collective, Indirect collective, Direct distributive, Indirect distributive

A discussion on this activity can be found at the end of the chapter.

Social interaction issues

There are an abundance of social issues which impact on the design of CSCW applications (Grudin, 1990). These extend through 3 main levels with an extensive variety of resulting issues:

- Communication: Verbal (e.g. language, intonation, tone, pauses) and non-verbal (e.g. body language such as gestures, eye contact) communications.
- Environmental and task factors: Factors associated with the specific environmental and task situation.
- Social, organisational and cultural norms: Norms associated with social groups, organisations, geographical and cultural boundaries.

To understand group collaboration at all these levels we must understand that social interaction is reliant on accepted norms of behaviour.

Mental models

Throughout our lives we develop mental models (internalised mental representations) of the world and our interactions. These representations play a central role in our understanding of the world, enabling us to predict and interact with it. You may have a mental model of a restaurant; what is expected to be found (e.g. plates, food, tables, cutlery), to occur (e.g. ordering food, eating food, paying for that food) and how you should behave there (e.g. etiquette - table manners, tipping etc). Mental models are not purely cognitive as they develop within cultures where learning them is part of our assimilation and socialisation. These mental models are, therefore, affected by previous experiences that are often embedded within cultural contexts (e.g. US tipping culture, Greek plate throwing etc). This means that mental models are not scientifically based but instead incomplete, unstable and often superstitiously based.

In order for us to communicate with others it is important that there is a joint understanding with reference to our mental models. There are, therefore, socially determined mental models that we construct for both verbal (conversation, intonation, pauses) and non-verbal (e.g. body-language, eye contact) communications. Specific communication environments and tasks affect these mental models as well as the relevant social, organisational and cultural norms.

Social cues

We all assume that in many situations we know the codes of practice (have mental models) of what is acceptable and unacceptable behaviour (e.g. acceptable to clap at the end of a theatre performance but not at the end of a funeral service). However, these codes vary within different cultures and these cultures can vary between organisations, cities or countries. These differences in social codes can relate strongly to many collaboration systems as they cross many organisational, country and cultural boundaries. In the real world moving between different cultures can be difficult but the human ability to adapt enables many of us travelling between different cultures to learn what is acceptable or unacceptable from others within that culture. What is of vital importance, therefore, is for us to receive accurate feedback of what is acceptable and unacceptable within that culture for that specific environment and task. Social cues of norms & pressures are investigated to produce informed knowledge of these acceptable and unacceptable behaviours.

Every perception we have and every action we take is embedded within our experiences and understanding of the social world around us. Yet, social interaction is complex and, although researched for centuries, much of the reasoning behind our social behaviours still eludes us. It is not surprising then, that computer-mediated interactions often develop into a simplification of the real world. However, because users equate mediated life with real life, computer mediated interactions often trigger a wealth of socially determined responses, whether the system designers expected them to occur or not. Within CSCW systems real world metaphors are often used to assist and shape interactions. However, it must be understood that many everyday assumptions we make to help us navigate our everyday life are not supported and are inaccurate within CSCW systems.

The Internet, in particular, covers all continents and thus many cultures and yet it can isolate us from the very social cues that allow us to adapt our behaviours accordingly. Within the virtual world there are often no clear communities with cues from others of what is acceptable or unacceptable behaviour. Many collaborative and communication environments are designed around replicating real world spaces through spatial metaphors. However, these replications often do not produce the socially constructed understanding of place we require for mediating our interactions. Ultimately, many cues can be relayed within a virtual format but much is often missed. Worse than a lack of cues, however, is the presentation of inaccurate cues. If we have an inaccurate mental model of a communication environment we are likely to predict inaccurately its behaviour or act inappropriately.

Activity 3 – Verbal and non-verbal cues

Identify within a conversation some verbal cues and non-verbal cues we use to identify if someone is bored, angry or confused.

A discussion on this activity can be found at the end of the chapter.

Activity 4 - Cultural cues

Identify some variations between two different cultures for verbal and non-verbal cues

A discussion on this activity can be found at the end of the chapter.

CSCW impacts on social interaction

People need social cues about the type of situation in which they find themselves and the type of behaviour with which they should respond. We also use social cues to assess those with whom we are interacting and how they perceive us. CSCW systems and applications vary in the level of contextual cues provided that enable users to appropriately frame their interactive behaviour. Many of the cues that are provided distort the resultant interaction. The three levels of social interaction issues previously mentioned (communication, environment / task factors and social, organisational and cultural norms) are used to assess the type of misconceptions that can arise out of distorted or non-existent CSCW system cues.

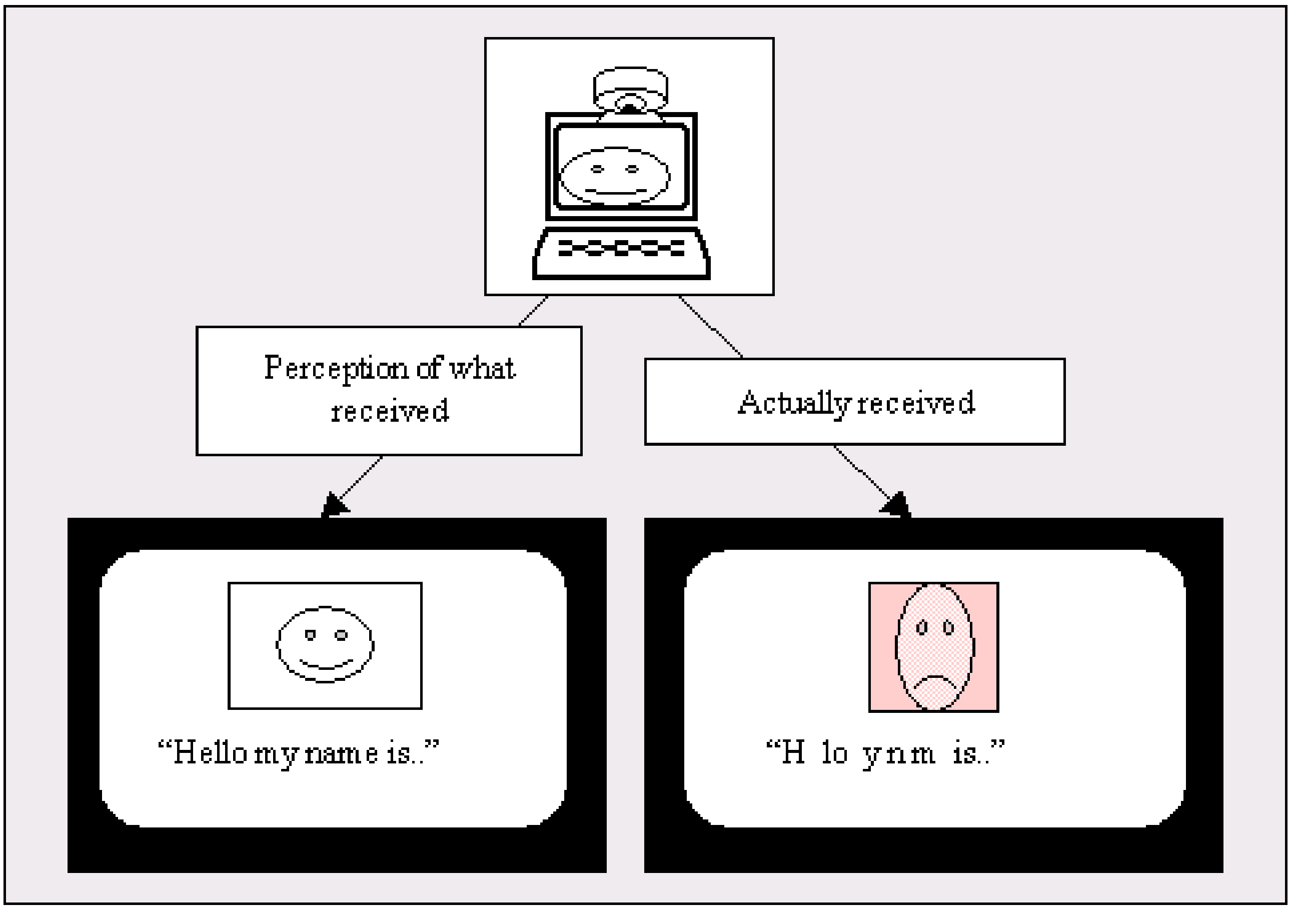
Communication

Non-verbal communication

Image quality, camera angles or lost data during transmission (due to packet loss) may result in a perception of the user that they regard as inaccurate. Distorted images can make people look bored, angry or upset when this is not the case. A lack of feedback to those releasing the data about how it looks when received may also produce inaccurate assumptions about the interaction.

Verbal communication

Due to information being lost when communicating, users can get a distorted perception of the interaction and the people they are interacting with. Audio loss can slow down the conversations and increase misinterpretations about the task.



Environmental and task factors

Temporal contexts

When interacting across temporal contexts as with asynchronous interactions the situational context (environment, conditions) may change during the task. This then means that some users may misinterpret the task they are collaborating on.

Task awareness

Lack of feedback in some CSCW systems, in synchronous interactions, can cause task state changes to go unnoticed or misinterpreted. This can cause a lack of task continuity for some of the users.

Integrating tools and work procedures

Collaboration becomes cumbersome when new tools have to be learned or used (causing data format translation problems) for the same task in collaborative mode from the ones used in non-collaborative mode (Schneiderman, 1998).

Social, organisational and cultural norms

Collaborators may not share the same assumptions and beliefs. Cultures can be national, geographical, ethnic, social, educational, and organisational.

Security and privacy

CSCW system design should ensure users awareness of interaction accessibility. There can be problems when users accidentally drift from private to public space/directory/conference, or if then are not aware what others can do with their data.

Cultural norms

A publicly transmitted interaction that implies, by contextual cues, to people from one culture that it is private but implies to someone from another culture that this is a public interaction is likely to lead to an invasion of the former users’ privacy.

Activity 5 – Unsupported non-verbal cues

Identify a non-verbal communication cue that could cause problems if not supported by a CSCW system. How would it cause problems?

Designing CSCW systems

Cooperation between individuals, groups and organisations is still increasing so the demand for Groupware/CSCW systems is likely to increase. It is therefore, vital that the design of these systems supports rather than impedes collaborative work (Grudin, 1990). To this end CSCW system designers must try to understand:

1. Multiple users and their social interaction
2. Collaborative work

Designing for multiple users and their social interactions

Multiple users increase the importance of users’ involvement in requirements capture and design. This can be done through user centred design procedures (see units 7, 8 and 9) and there are various other methodologies that allow for capturing the complexities involved in social interactions (participatory design, ethnography and grounded theory - a social science methodology recently applied by some CSCW researchers)

When designing CSCW systems it is important to understand the multiple roles that users have in the interaction and therefore not to design for users as one body e.g.

When designing CSCW systems it is important to understand the multiple roles that users have in the interaction and therefore not to design for users as one body e.g.

Feedback is an important element in CSCW system design. Ultimately there is a need for accurate contextualisation of data for all parties. The more appropriate social interaction feedback parties receive, the easier it will be to appropriately support social interactions e.g.

In the real world standing too close to someone or staring at them for too long would result in disapproving looks, coughs, sighs etc. A lack of the facial and body cues that we take for granted in real world situations can produce an isolating and inhibiting situation for a user.

Designing for collaborative work

Designing for collaborative work increases the importance of understanding all aspects of the collaborative task and potential state changes.

Keep it simple to start with, let users evaluate the system early on, and give them time and opportunity to learn how to use the system before using it for real tasks.

Integrate CSCW application with other tools users employ e.g. Make sure data from users tools can be easily incorporated into shared applications and vice versa - there is no greater waste of time than having to re-create or transform data.

Ensure interoperability with other users/systems/organizations.

Collaborative working carries overheads (time, cost, coordination, communication) as well as potential benefits (resource sharing, improved quality, cultural diversity). The trade-offs that users are prepared to make against these overheads must be understood. It is important to understand whether or not your CSCW system has lost the major benefit that could be traded off against what is considered minor inconveniences.

Activity 6 - CSCW application assessment

Review a CSCW application and identify if the system does not support required:

- Communication cues
- Environmental and task factors
- Social, organisational and cultural norms.

Review Question 7

Describe an interaction problem that may be caused by group members working asynchronously

Answer at the end of the chapter.

Review Question 8

Scenario: An office worker goes to his single occupant office after work to change for a squash match. Later he is told by a colleague that they liked the colour of his underpants which they saw on their monitor via the organisational awareness technology system.

Why did this scenario occur?

Answer at the end of the chapter.

Discussion Topics

Discussion 1

Discuss in which circumstances computer support for collaborative work is inappropriate and how you would make sure that CSCW is appropriate?

Comments on this discussion topic can be found at the end of the chapter.

Discussion 2

Discuss how Internet multimedia communication collaborations (video conferencing, virtual reality) may cause problems with social interactions between cultures.

Comments on this discussion topic can be found at the end of the chapter.

Answers and Discussions

Answer to Review Question 1

As international economies have risen in their importance so has the importance of international collaboration. The costs of international travel in time, money and physical exhaustion compares unfavourably with the alternative of installing and using CSCW tools and applications

Answer to Review Question 2

Previous concentration by disciplines such as HCI on one user one application support has hindered the development of multiple user applications and tools.

Answer to Review Question 3

Defining the group work which requires support Detailing which aspects of group working the CSCW system & software support Identifying what potential obstacles the technology may impose on group working.

Answer to Review Question 4

A formal group

Answer to Review Question 5

The rigid specific sub-task assigned project of writing a co-authored book requires group members to be able to work in a staggered manner working, for portions of their time, alone and non-collaboratively. An asynchronous application would therefore support these work practices. A brain-storming task, however, requires group members to respond immediately to collaborative participants and requires a synchronous application to support this form of cooperation.

Answer to Review Question 6

FALSE: This would be classified as a collective indirect (not direct) collaboration as the collaboration is mediated by the task changes and not the group members in direct interpersonal communication.

Answer to Review Question 7

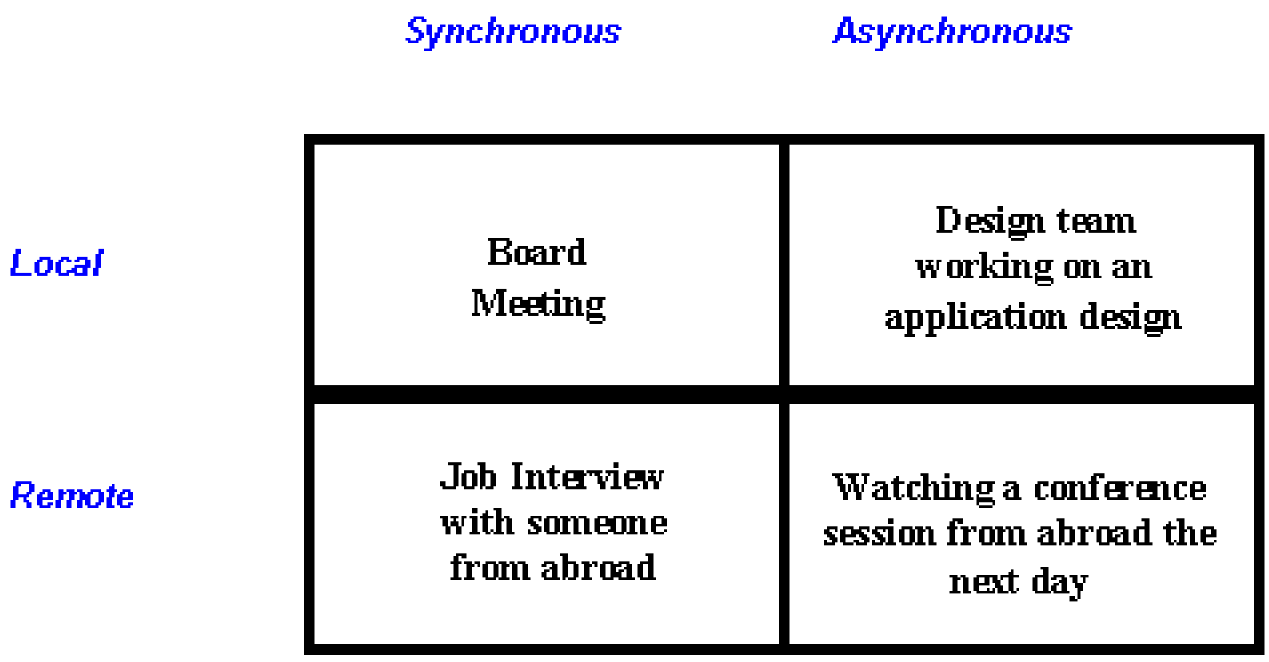
The task may change without some members being aware of it and thus these group members may be working on something already discarded as unimportant by the rest of the group.

Answer to Review Question 8

A lack of system feedback to the user that the system was still on, what other users could see made the office worker feel secure in his physical surroundings that the situation was private when it was not.

Discussion on Activity 1

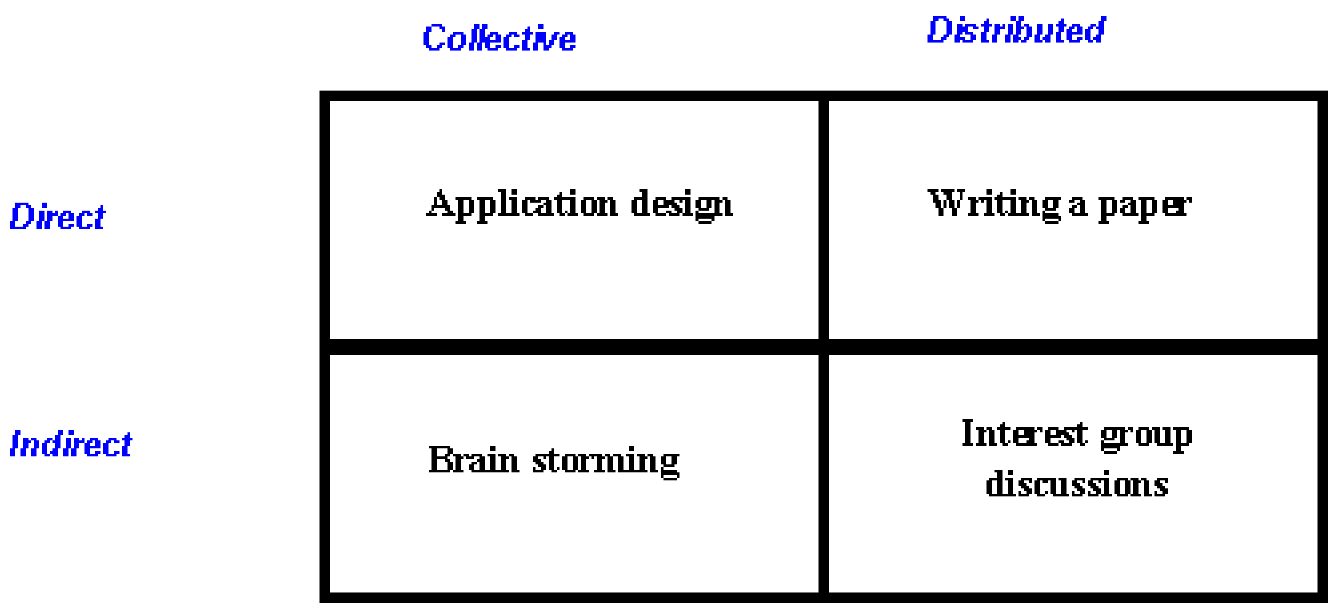
In identifying the tools and applications it may help to fill in a table (below is what Dix, et al. 1998 describe as a time and space matrix). Below are some examples of the tasks to be supported by the tools and applications.



	Synchronous	Asynchronous
Local	Board Meeting	Design team working on an application design
Remote	Job Interview with someone from abroad	Watching a conference session from abroad the next day

Discussion on Activity 2

In identifying the tools and applications it may help to fill in a table (below is what Dix, et al. 1998 describe as a time and space matrix). Below are some examples of the tasks to be supported by the tools and applications.



	Collective	Distributed
Direct	Application design	Writing a paper
Indirect	Brain storming	Interest group discussions

Discussion on Activity 3

To help identify these cues consider:

- With verbal cues it will help to think of intonation, tones, flow of the conversation
- With non-verbal cues it will help to think of body language, eye contact, actions

Discussion on Activity 4

To help identify these cues consider differences between cultures for:

- With verbal cues it will help to think of intonation, tones, flow of the conversation
- With non-verbal cues it will help to think of body language, eye contact, actions

Comments on Discussion 1

It is important to understand that technology is not appropriate for all situations and in some contexts may impede effective real-world interactions. It has recently been argued that people teleworking need to have occasional face-to-face meetings to ‘stay in touch’ with what is going on. Many of the reasons behind this need for real-world meetings relates to social interaction issues of trust, relationships etc.

Comments on Discussion 2

Within different cultures variations in social norms (e.g. privacy requirements, communication norms) can cause problems with communications. Interpersonal distance has been found to dictate the intensity of a response: faces in a close-up are scrutinised and attended to more often than those in the background. Yet, between cultures the personal space required for comfortable interactions varies. How would you allow for the right distance, in an application, to increase attention levels yet allow for variations in personal space?