CHAPTER 5

Interaction Design Basics

Overview

Interaction design is about creating interventions in often complex situations using technology of many kinds including PC software, the web and physical devices

- Design involves:
  - achieving goals within constraints and trade-off between these
  - understanding the raw materials: computer and human
  - accepting limitations of humans and of design
- The design process has several stages and is iterative and never complete
- Interaction starts with getting to know the users and their context:
  - Finding out who they are and what they are like … probably not like you!
  - Talking to them, watching them
- Scenarios are rich design stories, which can be used and reused throughout design
  - They help us see what users will want to do
  - They give a step-by-step walkthrough of users' interactions: including what they see, do and are thinking
- Users need to find their way around a system; this involves
  - Helping users know where they are, where they have been and what they can do next
  - Creating overall structures that are easy to understand and fit the users' needs
  - Designing comprehensible screens and control panels
- Complexity of design means we don't get it right first time:
  - So we need iteration and prototypes to try out and evaluate
  - But iteration can get trapped in *local maxima*, designs that have no simple improvements, but are not good
  - Theory and models can help give good start points

5.1 Introduction

Some of HCI is focused on understanding: the academic study of the way people interact with technology. However, a large part of HCI is about doing things and making things – design.

In this chapter we will think about interaction design. Note that we are not just thinking about the design of interactive systems, but about the interaction itself. An office has just got a new electric stapler. It is connected to the mains electricity and is hard to move around, so when you want to staple papers together you go to the stapler. In the past when someone wanted to staple things they would take the stapler to their desk and keep it until someone else wanted it. You might write a letter, print it, staple it, write the next, staple it ... Now you have to take the letters to be stapled across the office, so instead you write–print, write–print until you have a pile of things to staple and then take it across. The stapler influences the whole pattern of interaction.

So, interaction design is not just about the artefact that is produced, whether a physical device or a computer program, but about understanding and choosing how that is going to affect the way people work. Furthermore, the artefacts we give to people are not just these devices and programs, but also manuals, tutorials, on-line help systems. In some cases we may realize that no additional system is required at all; we may simply suggest a different way of using existing tools.

Because of this it may be better not to think of designing a system, or an artefact, but to think instead about *designing interventions*. The product of a design exercise is that we intervene to change the situation as it is; we hope, of course, changing it for the better!

In the next section we will ask “what is design?” which sets the spirit for the rest of the chapter. Section 5.3 looks at the design process as a whole and this gives a framework for the following sections. Section 5.4 looks at aspects of the requirements gathering phase of design focused on getting to know and understand the user. This is followed in section 5.5 by a look at scenarios, which are a way of recording existing situations and examining proposed designs. We then look at the details of designing the overall application structure in Sections 5.6 and individual screen design in Section 5.7. Because design is never perfect first time (or ever!), most interaction design involves several cycles of prototyping and evaluation. The chapter ends with an examination of the limits of this and why this emphasizes the importance of deep knowledge of more general theories and models of interaction.

This chapter also functions as an introduction to much of Part 2 and Part 3 of this book. In particular, Section 5.3 puts many of the succeeding chapters into the context of the overall design process. Many of the individual sections of this chapter give early views, or simple techniques, for issues and areas dealt with in detail later in the book.

5.2 What is design?

So what is design?
A simple definition is:

achieving goals within constraints

This does not capture everything about design, but helps to focus us on certain things:

- *goals* – What is the purpose of the design we are intending to produce? Who is it for? Why do they want it? For example, if we are designing a wireless personal movie player, we may think about young affluent users wanting to watch the latest movies whilst on the move and download free copies, and perhaps wanting to share this with a few friends.
- *constraints* – What materials must we use? What standards must we adopt? How much can it cost? How much time do we have to develop it? Are there health and safety issues? In the case of the personal movie player: does it have to withstand rain? Must we use existing video standards to download movies? Do we need to build in copyright protection?

Of course, we cannot always achieve all our goals within the constraints. So perhaps one of the most important thing about design is:

- *trade-off* – Choosing which goals or constraints can be relaxed so that others can be met. For example, we might find that an eye-mounted video display, a bit like those used in virtual reality, would give the most stable image whilst walking along. However, this would not allow you to show friends, and might be dangerous if you were watching a gripping part of the movie as you crossed the road.

Often the most exciting moments in design are when you get a radically different idea that allows you to satisfy several apparently contradictory constraints. However, the more common skill needed in design is to accept the conflict and choose the most appropriate trade-off. The temptation is to focus on one or other goal and optimize for this, then tweak the design to make it just satisfy the constraints and other goals. Instead, the best designs are where the designer understands the trade-offs and the factors affecting them. Paradoxically, if you focus on the trade-off itself the more radical solutions also become more apparent.

The golden rule of design

Part of the understanding we need is about the circumstances and context of the particular design problem. We will return to this later in the chapter. However, there are also more generic concepts to understand. The designs we produce may be different, but often the raw materials are the same. This leads us to the *golden rule of design:*

understand your materials

In the case of a physical design this is obvious. Look at a chair with a steel frame and one with a wooden frame. They are very different: often the steel frames are tubular or thin L or H section steel. In contrast wooden chairs have thicker solid legs. If you made a wooden chair using the design for a metal one it would break; if you made the metal one in the design for the wooden one it would be too heavy to move.

For Human–Computer Interaction the obvious materials are the human and the computer. That is we must:

- understand *computers*
  - limitations, capacities, tools, platforms
- understand *people*
  - psychological, social aspects, human error

Of course, this is exactly the focus of chapters 1 and 2. This is why they came first; we must understand the fundamental materials of human–computer interaction in order to design. Of course in chapters 3 and 4 we also looked at the nature of *interaction* itself. This is equally important in other design areas. For example, the way you fit seats and windows into an airplane's hull affects the safety and strength of the aircraft as a whole.

To err is human

It might sound demeaning to regard people as 'materials', possibly even dehumanizing. In fact, the opposite is the case; physical materials are treated better in most designs than people. This is particularly obvious when it comes to failures.

The news headlines: an aircrash claims a hundred lives, an industrial accident causes millions of pounds’ worth of damage, the discovery of systematic mistreatment leads to thousands of patients being recalled to hospital. Some months later the public inquiry concludes: human error in the operation of technical instruments. The phrase ‘human error’ is taken to mean ‘operator error’, but more often than not the disaster is inherent in the design or installation of the human interface. Bad interfaces are slow or error prone to use. Bad interfaces cost money and cost lives.

People make mistakes. This is not ‘human error’, an excuse to hide behind in accident reports, it is human nature. We are not infallible consistent creatures, but often make slips, errors and omissions. A concrete lintel breaks and a building collapses. Do the headlines read ‘lintel error’? No. It is the nature of concrete lintels to break if they are put under stress and it is the responsibility of architect and engineer to ensure that a building only puts acceptable stress on the lintel. Similarly it is the nature of humans to make mistakes and systems should be designed to reduce the likelihood of those mistakes and to minimize the consequences when mistakes happen.

Often when an aspect of an interface is obscure and unclear, the response is to add another line in the manual. People are remarkably adaptable and, unlike concrete lintels, can get ‘stronger’, but better training and documentation (although necessary) are not a panacea. Under stress, arcane or inconsistent interfaces will lead to errors.

If you design using a physical material, you need to understand how and where failures would occur and strengthen the construction, build in safety features or redundancy. Similarly, if you treat the human with as much consideration as a piece of steel or concrete, it is obvious that you need to understand the way human failures occur and build the rest of the interface accordingly.

The central message – the user

In this book you will find information on basic psychology, on particular technologies, on methods and models. However, there is one factor that outweighs all this knowledge. It is about attitude. Often it is said that the success of the various methods used in HCI lies not in how good they are, but in that they simply focus the mind of the designer on the user.

This is the core of interaction design: put the user first, keep the user in the centre and remember the user at the end.

5.3 The process of design

Often HCI professionals complain that they are called in too late. A system has been designed and built, and only when it proves unusable do they think to ask how to do it right! In other companies usability is seen as equivalent to testing – checking whether people can use it and fixing problems, rather than making sure they can from the beginning. In the best companies, however, usability is designed in from the start.

In chapter 6 we will look in detail at the software development process and how HCI fits within it. Here we’ll take a simplified view of four main phases, focused on the design of interaction (figure 5.1).

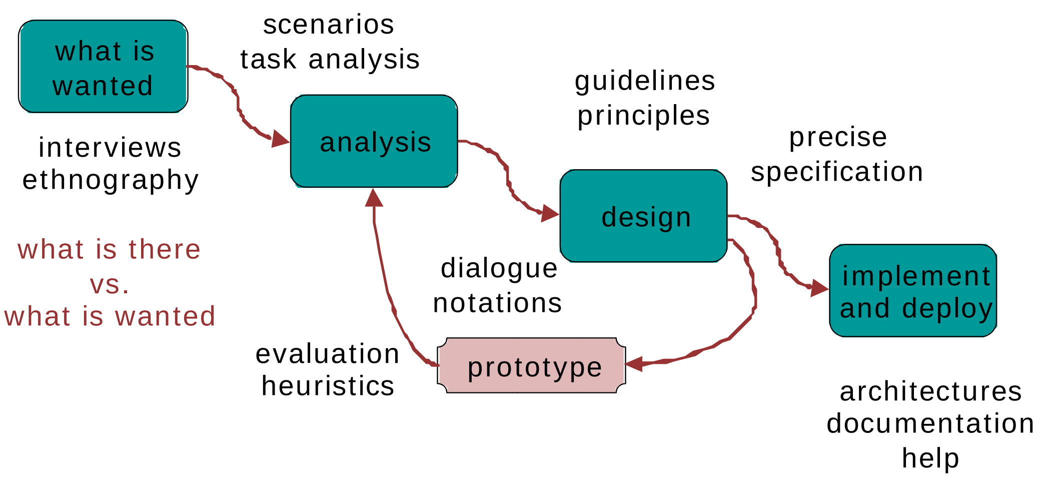


Figure 5.1 Interaction design process

requirements – what is wanted

The first stage is establishing what exactly is needed. As a precursor to this it is usually necessary to find out what is currently happening. For example, how do people currently watch movies? What sort of personal appliances do they currently use?

There are a number of techniques used for this in HCI: interviewing people, videotaping them, looking at the documents and objects that they work with, observing them directly. We don't have a chapter dedicated to this, but aspects of this will be found in various places. In particular, ethnography, a form of observation deriving from anthropology, has become very influential and is discussed in chapter 13. We will look at some ways of addressing this stage in section 5.4.

analysis

The results of observation and interview need to be ordered in some way to bring out key issues and communicate with later stages of design. Chapter 15 and part of chapter 18 deal with task models, which are a means to capture how people carry out the various tasks that are part of their work and life. In this chapter (section 5.5), we will look at scenarios, rich stories of interaction, which can be used in conjunction with a method like task analysis or on their own to record and make vivid actual interaction. These techniques can be used both to represent the situation as it is and also the desired situation.

design

Well, it is all about design, but there is a central stage when you move from what you want to how to do it. There are numerous rules, guidelines and design principles that can be used to help with this and Chapter 7 discusses these in detail; chapter 10 looks at how to design taking into account many different kinds of user. We need to record our design choices in some way and there are various notations and methods to do this, including those used to record the existing situation. Chapters 16, 17 and 18 deal with ways of modelling and describing interaction. In this chapter, section 5.6 will look at some simple notations for designing navigation within a system and some basic heuristics to guide the design of that navigation. Section 5.7 will look more closely at the layout of individual screens. It is at this stage also where input from theoretical work is most helpful, including cognitive models, organizational issues and understanding communication (Chapters 12, 13 and 14).

iteration and prototyping

Humans are complex and we cannot expect to get designs right first time. We therefore need to evaluate a design to see how well it is working and where there can be improvements. We will discuss some techniques for evaluation in Chapter 9. Some forms of evaluation can be done using the design on paper, but it is hard to get real feedback without trying it out. Most user interface design therefore involves some form of prototyping, producing early versions of systems to try out with real users. We'll discuss this in section 5.8.

implementation and deployment

Finally, when we are happy with our design we need to create it and deploy it. This will involve writing code, perhaps making hardware, writing documentation and manuals – everything that goes into a real system that can be given to others. Chapter 8 will deal with software architectures for user interfaces and there are details about implementing groupware in chapter 19 and web interfaces in chapter 21.

If you read all the chapters and look at all the techniques you might think "help! how can I ever do all this?". Of course the answer is you can't. Your time is limited – there is a trade-off between the length of the design period and the quality of the final design. This means one sometimes has to accept a design as final even if it is not perfect: it is often better to have a product that is acceptable but on time and to cost than it is to have one that has perfect interaction but is late and over budget.

It is easy to think that the goal, especially of the iterative stages, is to find usability problems and fix them. As you experience real designs, however, you soon find that the real problem is not to find problems – that is easy; nor to work out how to fix them – that may not be too difficult; instead the issue is: which usability problems is it worth fixing?

In fact, if you ever come across a system that seems to be perfect it is a badly designed system – badly designed not because the design is bad, but because too much effort will have been spent in the design process itself. Just as with all trade-offs it may be possible to find radically different solutions that have a major effect but are cheap to implement. However, it is best not to plan assuming such bolts of inspiration will strike when wanted!

5.4 User focus

As we've already said, the start of any interaction design exercise must be the intended user or users. This is often stated as:

know your users

Because this sounds somewhat like a commandment it is sometimes even written "know thy user" (and originally "know the user" [[H71]]). Note, too, a little indecision about user/users – much of traditional user interface design has focused on a single user. We will discuss issues of collaboration extensively in Chapters 13 and 19, but even at this stage it is important to be aware that there is rarely one user of a system. This doesn't mean that every system is explicitly supporting collaboration like email does. However, almost every system has an impact beyond the person immediately using it.

Think about a stock control system. The warehouse manager queries the system to find out how many six-inch nails are in stock – just a single user? Why did he do this? Perhaps a sales person has been asked to deliver 100,000 six-inch nails within a fortnight and wants to know if the company is able to fulfill the order in time. So the act of looking at the stock control system involves the warehouse manager, the sales person and the client. The auditors want to produce a valuation of company assets including stock in hand, the assistant warehouse manager needs to update the stock levels while his boss is on holiday.

Over time many people are affected directly or indirectly by a system and these people are called *stakeholders* (see also chapter 13). Obviously tracing the tenuous links between people could go on forever and you need to draw boundaries as to whom you should consider. This depends very much on the nature of the systems being designed, but largely requires plain common sense.

So, how do you get to know your users?

- • who are they?

Of course, the first thing to find out is who your users are. Are they young or old, experienced computer users or novices? As we saw with the stock control system, it may not be obvious who the users are, so you may need to ask this question again as you find out more about the system and its context. This question becomes harder to answer if you are designing generic software, such as a word processor, as there are many different users with different purposes and characteristics. A similar problem arises with many web sites where the potential visitors are far from homogenous. It may be tempting to try to think of a generic user with generic skills and generic goals; however, it is probably better, either instead or in addition, to think of several specific users.

- • probably not like you!

When designing a system it is easy to design it as if *you* were the main user: you assume your own interests and abilities. So often you hear a designer say “but it’s obvious what to do”. It may be obvious for her! This is not helped by the fact that many software houses are primarily filled with male developers. Although individuals differ a lot there is a tendency for women to have better empathetic skills.

- • talk to them

It is hard to get yourself inside someone else’s head, so the best thing is usually to ask them. This can take many forms: structured interviews about their job or life, open-ended discussions, or bringing the potential users fully into the design process. The last of these is called *participatory design* (see Chapter 13, section 13.3.3). By involving users throughout the design process it is possible to get a deep knowledge of their work context and needs. The obvious effect of this is that it produces better designs. However, there is a second motivational effect, perhaps at least as important as the quality of the design. By being involved, users come to ‘own' the design and become champions for it once deployed. Recall that a system must be not only useful and usable, but also *used*.

People may also be able to tell you about how things *really* happen, not just how the organization says they *should* happen. To encourage users to tell you this, you will need to win their trust, since often the actual practices run counter to corporate policy. However it is typically these *ad hoc* methods that make organizations work, not the official story!

- • watch them

Although what people tell you is of the utmost importance, it is not the whole story.

When black-belt judo players are asked how they throw an opponent, their explanations do not match what they actually do. Think about walking – how do your legs and arms move? It is harder than you would think! Although people have run since the earliest times, it was only with Eadweard Muybridge’s pioneering time lapse photography in the 1870s that the way people actually walk, run and move became clear (see figure 5.2). This is even more problematic with intellectual activities as it is notoriously difficult to introspect.

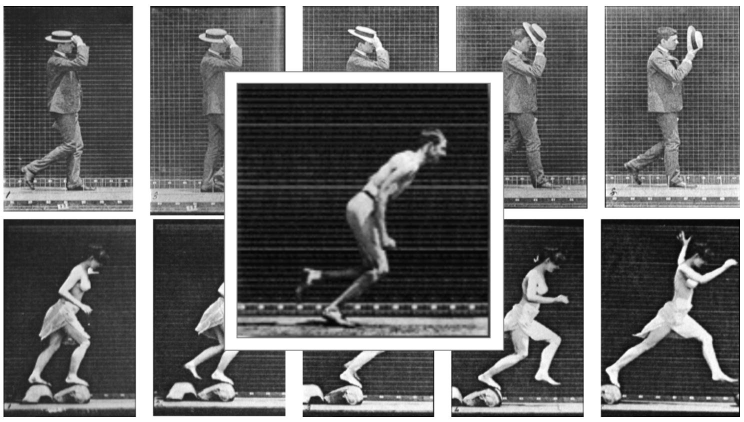


Figure 5.2 Eadweard Muybridge’s time lapse photography

A professional in any field is very practiced and can *do* things in the domain. An academic in the same field may not be able to do things, but she knows *about* the things in the domain. These are different kinds of knowledge and skill. Sometimes people know both, but not necessarily so. The best sports trainers may not be the best athletes, the best painters may not be the best critics.

Because of this it is important to watch what people do as well as hear what they say. This may involve sitting and taking notes of how they spend a day, watching particular activities, using a video camera or tape recorder. It can be done in an informal manner or using developed methods such as ethnography or contextual inquiry, which we will discuss in chapter 13.

Sometimes users can be involved in this; for example, asking them to keep a diary or having a 15-minute buzzer and asking them to write down what they are doing when the buzzer sounds. Although this sounds just like asking the users what they do, the structured format helps them give a more accurate answer.

DESIGN FOCUS

Cultural probes

Traditional ethnography has involved watching people and being present. There is always a disruptive effect when someone is watching, but in, say, an office after a while the ethnographer becomes ‘part of the wall paper’ and most activities happen as normal. However, in some environments, for example the home or psychiatric patients, it is hard to go and watch people for long periods or at all. Cultural probes have been used as one way to gather rich views of an area without intrusion. These were originally developed as prompts for design [[GDP99]], but have also been adopted as an added method for ethnography [[HCRCR02]].

Cultural probes are small packs of items designed to provoke and record comments in various ways. They are given to people to take away and to open and use in their own environment. For example one probe pack for the domestic environment included a glass with a paper sleeve. You used the glass to listen to things and then wrote down what you heard. The same probe pack contained a repackaged disposable camera and a small solid-state voice recorder. When the packs are returned the notes, recordings, photos, etc. are used as a means of understanding what is significant and important for the people in the enviornment and as a means of enculturing designers.

For more see /e3/online/cultural-probes

http://www.crd.rca.ac.uk/equator/domestic_probes.html

.

Another way to find out what people are doing is to look at the artefacts they are using and creating. Look at a typical desk in an office. There are papers, letters, files, perhaps a stapler, a computer, sticky notes ... Some of these carry information, but if they were only important for the information in them they could equally well be in the filing cabinet and just taken out when needed. The post-it note on the edge of Brian's screen saying "book table" is not just information that he needs to book a restaurant table. The fact that it is on his screen is reminding him that something needs to be done. In Chapter 18 we will look at the role of artefacts in detail.

In all these observational methods one should not just stop at the observation, but go back and discuss the observations with the users. Even if they were not previously aware of what they do they are likely to be able to explain when shown. The observations tell you *what* they do, they will tell you *why*.

- • use your imagination

Even if you would like to involve many users throughout your design exercise it is not always possible. It may be too costly, it may be hard to get time with them (e.g. hospital consultant), it may be that there are just too many (e.g. the web). However, even if you cannot use actual users you can at least try to imagine what it is like for them.

Now this is very dangerous! It would be easy to think, "if I were a warehouse manager I would do this". The issue is not what *you* would do in the user's shoes but what *they* would do. This requires almost a kind of method acting. Imagine being a warehouse manager. What does the word "undo" in the menu mean to him?

One method that has been quite successful in helping design teams produce user focused designs is the *persona*. A persona is a rich picture of an imaginary person who represents your core user group. Figure 5.3 gives an example persona of Betty the warehouse manager. A design team will have several of these personae covering different types of intended users and different roles. The personae will themselves be based on studies of actual users, observation, etc. When a design solution is proposed the team can ask, "How would Betty react to this?". The detail is deliberately more than is strictly necessary, but this is essential. It is only by really feeling Betty is a real person that the team can start to imagine how she will behave.

Betty is 37 years old, She has been Warehouse Manager for five years and worked for Simpkins Brothers Engineering for twelve years. She didn't go to university, but has studied in her evenings for a business diploma. She has two children aged 15 and 7 and does not like to work late. She did part of an introductory in-house computer course some years ago, but it was interupted when she was promoted and could no longer afford to take the time. Her vision is perfect, but her right-hand movement is slightly restricted following an industrial accident 3 years ago. She is enthusiastic about her work and is happy to delegate responsibility and take suggestions from her staff. However, she does feel threatened by the introduction of yet another new computer system (the third in her time at SBE).

Figure 5.3 Persona – a rich description of Betty the Warehouse Manager

5.5 Scenarios

Scenarios are stories for design: rich stories of interaction. They are perhaps the simplest design representation, but one of the most flexible and powerful. Some scenarios are quite short: "the user intends to press the 'save' button, but accidentally presses the 'quit' button so loses his work". Others are focused more on describing the situation or context.

Figure 5.4 gives an example of a scenario for the personal movie player. Like the persona it is perhaps more detailed than appears necessary, but the detail helps make the events seem real. The figure shows plain text, but scenarios can be augmented by sketches, simulated screen shots, etc. These sketches and pictures are called *storyboards* and are similar to the techniques used in film-making to envisage plot-lines.

Brian would like to see the new film "Moments of Significance" and wants to invite Alsion, but he knows she doesn't like "arty" films. He decides to take a look at it to see if she would like it and so connects to one of the movie sharing networks. He uses his work machine as it has a higher bandwidth connection, but feels a bit guilty. He knows he will be getting an illegal copy of the film, but decides it is OK as he is intending to go to the cinema to watch it. After it downloads to his machine he takes out his new personal movie player. He presses the 'menu' button and on the small LCD screen he scrolls using the arrow keys to 'bluetooth connect' and presses the select button. On his computer the movie download program now has an icon showing that it has recognised a compatable device and he drags the icon of the film over the icon for the player. On the player the LCD screen says "downloading now", a percent done indicator and small wirling icon.

During lunch time he takes out his movie player, plugs in his earphones and starts to watch. He uses the arrow keys to skip between portions of the film and decides that yes, Alison would like it. Then he feels a tap on his shoulder. he turns rtound. it is Alison. He had been so absorbed he didn’t notice her. “What are you watching” she says. “Here, listen” he says and flicks a small switch. The built-in directional speaker is loud enough for both Brian and Alison to hear, but not loud enough to disturb other people in the canteen. Alison recognises it from trailers “surprised this is out yet” she says. “well actually ...” Brian confeses, “you’d better come with me to see it and make an honest man of me”. “I’ll think about it” she replies.

Figure 5.4 Scenario for proposed movie player

Where the design includes physical artefacts the scenarios can be used as a script to act out potential patterns of use. For example, we might imagine a digital Swiss Army Knife, which has a small LCD screen and uses the toothpick as a stylus. The knife connects to the Internet via a wireless link through your phone and gives interesting tips from other Swiss Army Knife users. Try getting two together at a party – you will see this would appeal! It sounds like a great design idea – but wait, try acting out the use. If you have a Swiss Army Knife, use it, or use something penknife-sized if you don’t. The tip on the LCD says, “open the stone remover” – a small LED glows near the right blade – you open it. “Now push the blade into the rubber of the grommet”, it says. You do this and then look for the next instruction. Look at the knife in your hand ... oops, your thumb is covering where the screen would be. Perhaps a voice interface would be better.

You can see already how scenarios force you to think about the design in detail and notice potential problems before they happen. If you add more detail you can get to a blow-by-blow account of the user–system interactions and then ask ‘what is the user intending now”, “what is the system doing now”. This can help to verify that the design would make sense to the user and also that proposed implementation architectures would work.

In addition scenarios can be used to:

- **communicate with others** – other designers, clients or users. It is easy to misunderstand each other whilst discussing abstract ideas. Concrete examples of use are far easier to share.
- **validate other models** – A detailed scenario can be ‘played’ against various more formal representations such as task models (discussed in chapter 15) or dialogue and navigation models (chapter 16 and below).
- **express dynamics** – Individual screen shots and pictures give you a sense of what a system would look like, but not how it behaves.

In the next section we will discuss ways of describing the patterns of interaction with a system. These are more complex and involve networks or hierarchies. In contrast scenarios are linear – they represent a single path amongst all the potential interactions.

This linearity has both positive and negative points:

- **time is linear** – Our lives are linear as we live in time and so we find it easier to understand simple linear narratives. We are natural storytellers and story listeners.

- **but no alternatives** – Real interactions have choices, some made by people, some by systems. A simple scenario does not show these alternative paths. In particular, it is easy to miss the unintended things a person may do.

Scenarios are a resource that can be used and reused throughout the design process: helping us see what is wanted, suggesting how users will deal with the potential design, checking that proposed implementations will work, and generating test cases for final evaluation.

More examples of scenarios at: `/e3/online/scenario`

5.6 Navigation design

As we stressed, the object of design is not just a computer system or device, but the socio-technical intervention as a whole. However, as design progresses we come to a point where we do need to consider these most tangible outputs of design.

Imagine yourself using a word processor. You will be doing this in some particular social and physical setting, for a purpose. However, now we are focusing on the computer system itself. You interact with at several levels:

- **widgets** – The appropriate choice of widgets and wording in menus and buttons will help you know how to use them for a particular selection or action.
- **screens or windows** – You need to find things on the screen, understand the logical grouping of buttons.
- **navigation within the application** – You need to be able to understand what will happen when a button is pressed, to understand where you are in the interaction.
- **environment** – The word processor has to read documents from disk, perhaps some are on remote networks. You swap between applications, perhaps cut and paste.

You can see similar levels in other types of application and device:

PC application	**web site**	**physical device**
widgets	form elements, tags and links	buttons. dials, lights, displays
screen design	page design	physical layout
navigation design	site structure	main modes of device
other apps and operating system	the web, browser, external links	the real world!

There are differences; for example, in the web we have less control of how people enter a site and on a physical device we have the same layout of buttons and dispalys no matter what the internal state (although we may treat them differently).

We discussed GUI widgets in chapter 3 and in the next section we will look at details of screen design. In this section we will look mainly at navigation design – that is the main screens or modes within a system and how they interconnect. However, we will also briefly consider how this interacts with the wider environment.

Just in case you haven't already got the idea, the place to start when considering the structure of an application is to think about actual use:

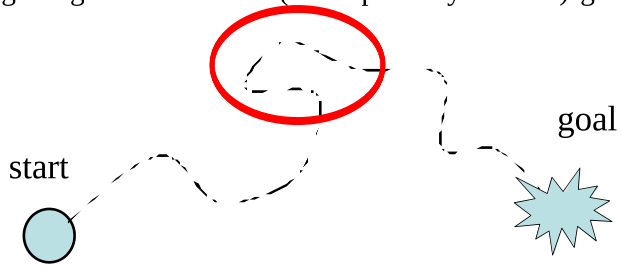
- who is going to use the application?
- how do they think about it?
- what will they do with it?

This can then drive the second task – thinking about structure. Individual screens or the layout of devices will have their own structure, but this is for the next section. Here we will consider two main kinds of issue:

- local structure
  - looking from one screen or page out
- global structure
  - structure of site, movement between screens

5.6.1 local structure

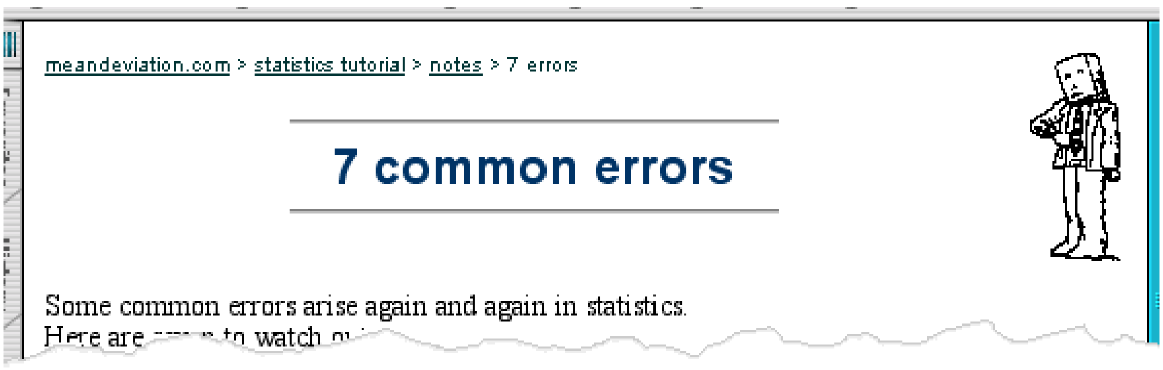
Much of interaction involves goal-seeking behaviour. Users have some idea of what they are after and a partial model of the system. In an ideal world if users had perfect knowledge of what they wanted and how the system worked they could simply take the shortest path to what they want, pressing all the right buttons and links. However, in a world of partial knowledge users meander through the system. The important thing is not so much that they take the most efficient route, but that at each point in the interaction they can make some assessment of whether they are getting closer to their (often partially formed) goal.



To do this goal seeking each state of the system or each screen needs to give the user enough knowledge of what to do to get closer to their goal. In Chapter 7 we will look at various design rules, some of which address this issue. To get you started here are four things to look for when looking at a single web page, screen or state of a device.

- knowing where you are
- knowing what you can do
- knowing where you are going – or what will happen
- knowing where you've been – or what you've done

The screen, web page or device displays should make clear *where you are* in terms of the interaction or state of the system. Some web sites show "bread crumbs" at the top of the screen, the path of titles showing where the page is in the site (figure 5.5). Similarly, in the scenario in Figure 5.4, the personal movie player says "downloading now", so Brian knows that it is in the middle of downloading a movie from the PC.


meandeviation.com > statistics tutorial > notes > 7 errors

7 common errors

Some common errors arise again and again in statistics.
Here are

Figure 5.5 Breadcrumbs

It is also important to know *what you can do* – what can be pressed or clicked to go somewhere or do something. Some web pages are particularly bad for this as it is unclear which images are pure decoration and which are links to take you somewhere.

On the web the standard underlined links make it clear which text is clickable and which is not. However, in order to improve the appearance of the page many sites change the colour of links and may remove the underline too. This is especially confusing if underline is then used as simple emphasis on words that are not links! The trade-off between appearance and ease of use may mean that this is the right thing to do, but you should take care before confusing the user needlessly!

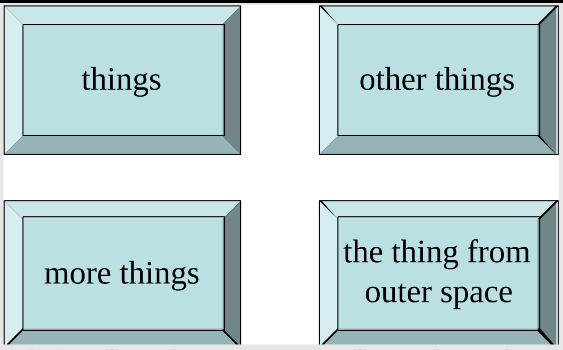
Chic design is also a problem in physical devices. One of the authors was once in a really high class hotel and found he could not locate the flush in the toilet. Only after much fumbling did he discover that one of the tiles could be pressed. The ‘active’ tile was level with the rest of the tiled wall – a very clean design, but not very usable!

You then need to know *where you are going* when you click a button or *what will happen*. Of course you can try clicking the button to see. In the case of a web site or information system this may mean you then have to use some sort of ‘back’ mechanism to return, but that is all; however, in an application or device the action of clicking the button may already have caused some effect. If the system has an easy means to undo or reverse actions this is not so bad, but it is better if users do not have to use this “try it and see” interaction. Where response times are slow this is particularly annoying.

Remember too that icons are typically not self-explanatory and should always be accompanied by labels or at very least tool tips or some similar technique. A picture paints a thousand words, but typically only when explained first using fifteen hundred!

Design Focus – Beware the big button trap

Public information systems often have touch screens and so have large buttons. Watch someone using one of these and how often they go to the wrong screen and have to use ‘back or ‘home’ to try again. If you look more closely you will find that each button has only one or two words on it giving the title of the next screen and possibly some sort of icon. Quite rightly the button label will be in a large font as users may have poor eyesight.



It is hard to choose appropriate labels that mean the same for everyone, especially when the breadth of the screen hierarchy is fixed by the maximum number of buttons. So it is no wonder that people get confused. However, there is usually plenty of room for additional explanation in a smaller font, possibly just the next level of button labels, or a sentence of explanation. It may not look as pretty, but it may means that people actually find the information they are looking for.

Special care has to be taken if the same command or button press means something different in different contexts. These different contexts that change the interpretation of commands are called *modes*. Many older text editors would interpret pressing ‘x’ to mean “enter me into the text” in a normal typing mode, but “exit” in a special command mode. If modes are clearly visible or audible this is less of a problem and in Chapter 3 (section 3.6.7) we saw how palettes are one way to achieve this. In general, modes are less of a problem in windowed systems where the mode is made apparent by the current window (if you remember which it is). However, physical devices may have minimal displays and may be operated without visual attention.

Finally, if you have just done some major action you also want some sort of confirmation of *what you've done*. If you are faultless and have perfect knowledge, of course you will be sure that you have hit the right key and know exactly what will happen. Remember too that to know what will happen you would need to know everything about the internal state of the system and things outside, like the contents of files, networked devices, etc., that could affect it. In other words, if you were omniscient you could do it. For lesser mortals the system needs to give some *feedback* to say what has happened.

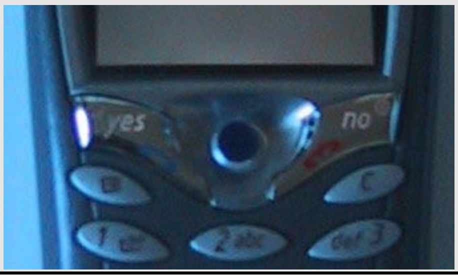
In an information system there is a related but slightly different issue, which is to know *where you have been*. This helps you to feel in control and understand your navigation of the information space. The feeling of disorientation when you do not have sufficient means to know where you are and where you have been has been called “lost in hyperspace”. Most web browsers offer a history system and also a ‘back’ button that keeps a list of recently visited pages.

Design Focus – modes

Alan’s mobile phone has a lock feature to prevent accidental use. To remove the lock he has to press the “C” (cancel) button which then asks for an additional “yes” to confirm removing the lock. So, in ‘locked’ mode “C” followed by “yes” means “turn off lock” and these are the most frequent actions when Alan takes the phone from his pocket.

However, Alan is forgetful and sometimes puts the phone in his pocket unlocked. This leads to occasional embarrassing phone calls and also to another problem.

The “yes” button is quite big and so this is often pressed while in his pocket. This puts the phone into “dial recent numbers” mode with a list of recent calls on screen. In this mode, pressing “C” gives a prompt “Delete number” and pressing “yes” then deletes the number from the phone’s adress book. Unhappily, this often means he takes the phone from his pocket, automatically presses “C”, “yes” only to see as he looks down to the handset the fatal words “number deleted”. Of course there is no undo!



5.6.2 global structure – hierarchical organisation

We will now look at the overall structure of an application. This is the way the various screens, pages or device states link to one another.

One way to organize a system is in some form of hierarchy. This is typically organized along functional boundaries (that is different kinds of things), but may be organized by roles, user type, or some more esoteric breakdown such as modules in an educational system.

The hierarchy links screens, pages or states in logical groupings. For example, figure 5.6 gives a high level breakdown of some sort of messaging system. This sort of hierarchy can be used purely to help during design, but can also be used to structure the actual system. For example, this may reflect the menu structure of a PC application or the site structure on the web.

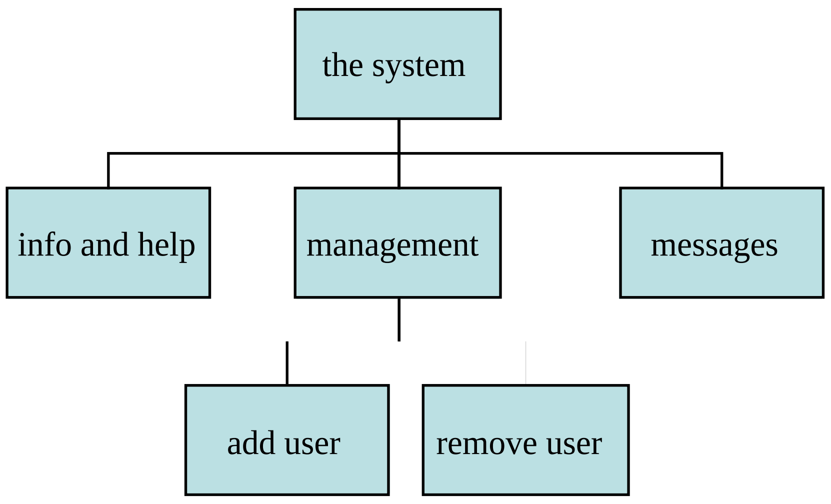


Figure 5.6 Application functional hierarchy

Any sort of information structuring is difficult, but there is evidence that people find hierarchies simpler than most. On of the difficulties with organizing information or system functionality is that different people have different internal structures for their knowledge, and may use different vocabulary. This is one of the places where a detailed knowledge of the intended users is essential: it is no good creating a hierarchy that the designers understand, but not the users ... and all too commonly this is exactly what happens.

However much you think you have got the wording and categories right, because there are different users it is inevitable that not everyone will understand it perfectly. This is where clear guidance as suggested in section 5.6.1 (*knowing where you are going – or what will happen*) is essential, as well as the means to allow users to change their mind if they make the wrong decisions.

There is also evidence that deep hierarchies are difficult to navigate, so it is better to have broad top-level categories, or to present several levels of menu on one screen or web page. Miller's magic number of 7+/-2 for working memory capacity (see chapter 1, section 1.3.2) is often misused in this context. Many guidelines suggest that menu breadth, that is the number of choices available at each level in the menu, should be around seven. However, Miller’s result applies only to working memory, not visual search. In fact optimal breadth can be quite large, perhaps 60 or more items for a web index page if the items are organised in such a way that the eye can easily find the right one [[LC98]]. (see `/e3/online/menu-breadth/` for more on optimal menu breadth). Of course, to organize the items on the page requires further classification. However, here the critical thing is the naturalness of the classification, which itself may depend on the user’s purpose. For example, if the user wants to look up information on a particular city an alphabetical list of all city names would be fast, but for other purposes a list by region would be more appropriate.

5.6.3 global structure – dialogue

In a pure information system or static web site it may be sufficient to have a fully hierarchical structure, perhaps with next/previous links between items in the same group. However, for any system that involves doing things, constantly drilling down from one part of the hierarchy to another is very frustrating. Usually there are ways of getting more quickly from place to place. For example, in the stock control system there may be a way of going from a stock item to all orders outstanding on that item and then from an order to the purchase record for the customer who placed the order. These would each be in a very different part of a hierarchical view of the application, yet directly accessible from one another.

As well as these cross links in hierarchies, when you get down to detailed interactions, such as editing or deleting a record, there is obviously a flow of screens and commands that is not about hierarchy. In HCI the word ‘dialogue’ is used to refer to this pattern of interactions between the user and a system.

Consider the following fragment from a marriage service:

Minister: do you *name* take this woman …
Man: I do
Minister: do you *name* take this man …
Woman: I do
Minister: I now pronounce you man and wife

Notice this describes the general flow of the service, but has blanks for the names of the bride and groom. So it gives the pattern of the interaction between the parties, but is instantiated differently for each service. Human–computer dialogue is just the same; there are overall patterns of movement between main states of a device or windows in a PC application, but the details differ each time it is run.

Recall that scenarios gave just one path through the system. To describe a full system we need to take into account different paths through a system and loops where the system returns to the same screen. There are various ways to do this, and in Chapter 16 we will expand on the wedding example and look at several different types of dialogue model.

A simple way is to use a network diagram showing the principal states or screens linked together with arrows. This can:

- show what leads to what
- show what happens when
- include branches and loops
- be more task oriented than a hierarchy

Figure 5.7 shows a network diagram showing the main screens for adding or deleting a user from the messaging system in figure 5.6. The arrows show the general flow between the states. We can see that from the main screen we can get to either the ‘remove user’ screen or the ‘add user’ screen. This is presumably by selecting buttons or links, but the way these are shown we leave to detailed screen design. We can also see that from the ‘add user’ screen the system always returns to the main screen, but after the ‘remove user’ screen there is a further confirmation screen.

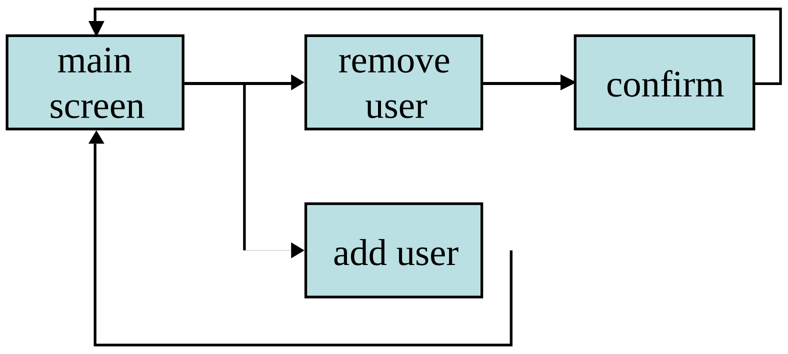


Figure 5.7 Network of screens/states

5.6.3 wider still

Donne said "No man is an Iland, intire of it selfe". This is also true of the things we design. Each sits amongst other devices and applications and this in turn has to be reflected within our design.

This has several implications:

- **style issues** – We should normally conform to platform standards, such as positions for menus on a PC application, to ensure consistency between applications. For example, on our proposed personal movie player we should make use of standard fast-forward, play, pause icons.
- **functional issues** – On a PC application we need to be able to interact with files, read standard formats, be able to handle cut and paste.
- **navigation issues** – We may need to support linkages between applications, for example allowing the embedding of data from one application in another, or, in a mail system, being able to double click an attachment icon and have the right application launched for the attachment.

On the web we have the added difficulty that other sites and applications may include links that bypass our 'home page' and other pages and go direct into the heart of our site or web application. Also, when we link to other sites, we have no control over them or the way their content may change over time.

5.7 Screen design and layout

We have talked about the different elements that make up interactive applications, but not about how we put them together. A single screen image often has to present information clearly and also act as the locus for interacting with the system. This is a complex area, involving some of the psychological understanding from Chapter 1 as well as aspects of graphical design.

The basic principles at the screen level reflect those in other areas of interaction design:

- **ask** – What is the user doing?

- **think** – What information is required? What comparisons may the user need to make? In what order are things likely to be needed?
- **design** – Form follows function: let the required interactions drive the layout.

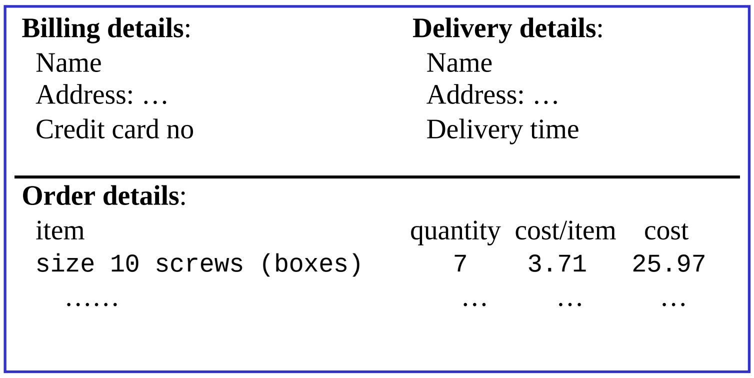
5.7.1 Tools for layout

We have a number of visual tools available to help us suggest to the user appropriate ways to read and interact with a screen or device.

Grouping and structure

If things logically belong together, then we should normally physically group them together. This may involve multiple levels of structure. For example, in figure 5.8 we can see a potential design for an ordering screen. Notice how the details for billing and delivery are grouped together spatially; also note how they are separated from the list of items actually ordered by a line as well as well as spatially. This reflects the following logical structure:

```
Order:
    Administrative information
        Billing details
        Delivery details
    Order information
        Order line 1
        Order line 2
        …
```



Billing details:
Name
Address: …
Credit card no

Delivery details:
Name
Address: …
Delivery time

Order details:

item	quantity	cost/item	cost
`size 10 screws (boxes)`	7	3.71	25.97
……	…	…	…

Figure 5.8 Grouping related items in an order screen

Order of groups and items

If we look at figure 5.8 again we can see that the fact the screen seems to naturally suggest reading or filling in the billing details first, followed by the delivery details, followed by the individual order items. Is this the right order?

In general we need to think: what is the natural order for the user? This should normally match the order on screen. For data entry forms or dialog boxes we should also set up the order in which the tab key moves between fields.

Occasionally we may also want to force a particular order; for example we may want to be sure that we do not forget the credit card details!

Decoration

Again, looking at figure 5.8 we can see how the design uses boxes and a separating line to make the grouping clear. Other decorative features like font style, and text or background colours can be used to emphasise groupings. Look at the microwave control panel in figure 5.9. See how the buttons differ in using the foreground and background colours (green and gold) so that groups are associated with one another. See also how the buttons are laid out to separate them into groups of similar function.

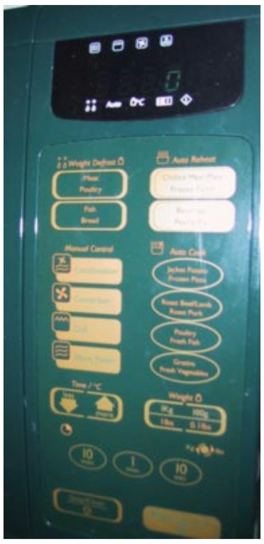


Figure 5.9 Microwave control panel

Alignment

Alignment of lists is also very important. For users who read text from left to right, lists of text items should normally be aligned to the left. Numbers, however, should normally be aligned to the right (for integers) or at the decimal point. This is because the shape of the column then gives an indication of magnitude – a sort of mini-histogram. Items like names are particularly difficult. Consider list (i) in Figure 5.10 . It is clearly hard to look someone up if you only know their surname. To make it easy, such lists should be laid out in columns as in (ii), or have forename and surname reversed as in (iii). (The dates in Figure 5.13, section 5.7.3, pose similar problems, as the years do not align, even when the folder is sorted by date.)

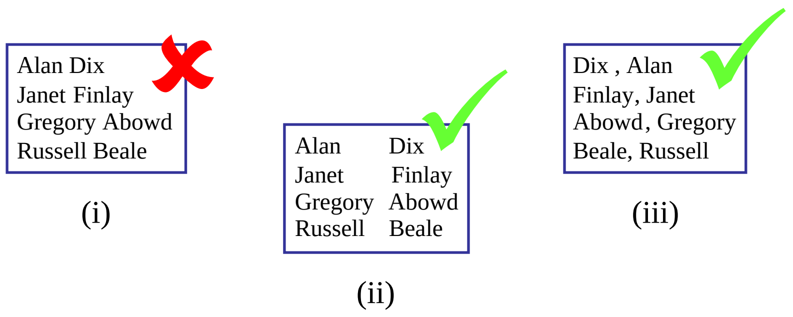
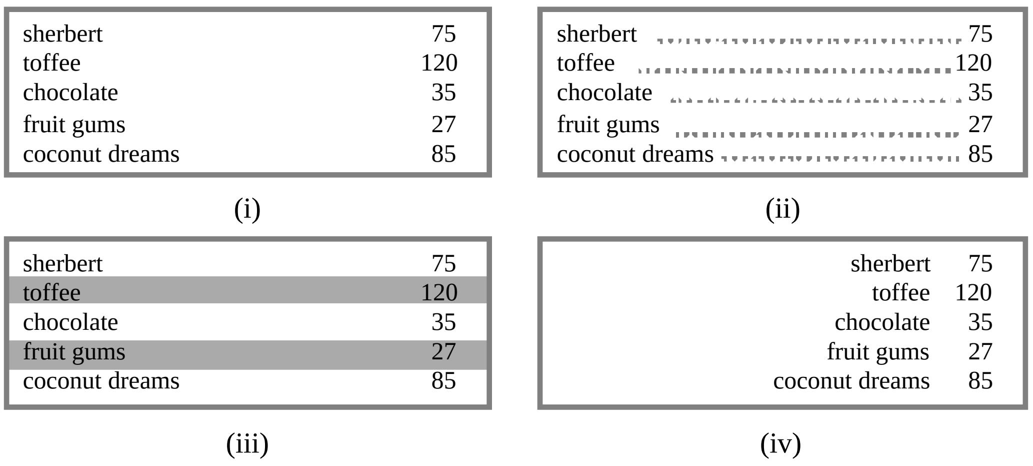


Figure 5.10 Looking up surnames

532.56	627.865
179.3	1.005763
256.317	382.583
15	2502.56
73.948	432.935
1035	2.0175
3.142	652.87
497.6256	56.34

Alignment and layout are important: find the biggest figure in each column

Multiple column lists require more care. Text columns have to be wide enough for the largest item, which means you can get large gaps between columns. Figure 5.11 shows an example of this (i) and you can see how hard this makes it for your eye to scan across the rows. There are several visual ways to deal with this including (ii) ‘leaders’ – lines of dots linking the columns – and (iii) using soft tone greys or colours behind rows or columns. It is also a time when it may be worth breaking other alignment rules, perhaps right aligning some text items as in (iv). This last alternative might be a good solution if you were frequently scanning the numbers and only occasionally scanning the names of items, but not if you needed to frequently look up names (which anyway are not sorted in this figure!). You can also see that this is an example of a design trade-off – good alignment for individual columns vs. ability to see relationship across rows.



(i)

sherbert	75
toffee	120
chocolate	35
fruit gums	27
coconut dreams	85

(ii)

sherbert	75
toffee	120
chocolate	35
fruit gums	27
coconut dreams	85

(iii)

sherbert	75
toffee	120
chocolate	35
fruit gums	27
coconut dreams	85

(iv)

sherbert	75
toffee	120
chocolate	35
fruit gums	27
coconut dreams	85

Figure 5.11 managing multiple columns

White space

In typography the space between the letters is called the counter. In painting this is also important and artists may focus as much on the space between the foreground elements such as figures and buildings as on the elements themselves. Often the shape of the counter is the most important part of the composition of a painting and in calligraphy and typography the balance of a word is determined by giving an even weight to the counters. If one ignores the ‘content’ of a screen and instead concentrates on the counter, the space between the elements, one can get an overall feel for the layout. If elements that are supposed to be related look separate when you focus on the counter, then something is wrong. Screwing up your eyes so that the screen becomes slightly blurred is another good technique for taking your attention away from the content and looking instead at the broad structure.

Space can be used to highlight

Space can be used in several ways. Some of these are shown in figure 5.12. The grey areas represent continuous areas of text or graphics. In (i) we can see space used to separate blocks as you often see in gaps between paragraphs or space between sections in a report. Space can also be used to create more complex structures. In (ii) there are clearly four main areas: ABC, D, E and F. Within one of these are three further areas, A, B and C, which themselves are grouped as A on its own, followed by B and C together. In figure 5.12 (iii), we can see space used to highlight. This is a technique used frequently in magazines to highlight a quote or graphic.

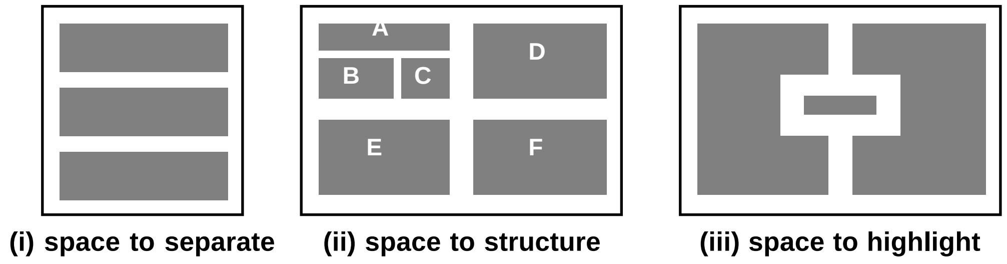


Figure 5.12 Using white space in layout

5.7.2 User action and control

Entering information

Some of the most complicated and difficult screen layouts are found in forms-based interfaces and dialog boxes. In each case the screen consists not only of information presented to the user, but also of places for the user to enter information or select options. Actually many of the same layout issues for data presentation also apply to fields for data entry. Alignment is still important. It is especially common to see the text entry boxes aligned in a jagged fashion because the field names are of different lengths. This is an occasion where right-justified text for the field labels may be best or, alternatively, in a graphical interface a smaller font can be used for field labels and the labels placed just above and to the left of the field they refer to.

For both presenting and entering information a clear logical layout is important. The task analysis techniques in Chapter 15 can help in determining how to group screen items and also the order in which users are likely to want to read them or fill them in. Knowing also that users are likely to read from left to right and top to bottom (depending on their native language!) means that a screen can be designed so that users encounter items in an appropriate order for the task at hand.

Knowing what to do

Some elements of a screen are passive, simply giving you information; others are active, expecting you to fill them in, or do something to them. It is often not even clear which elements are active, let alone what the effect is likely to be when you interact with them!

This is one of the reasons for platform and company style guides. If everyone designs buttons the same and menus the same, then users will be able to recognize them when they see them. However, this is not sufficient in itself. It is important that the labels and icons on menus are also clear. Again, standards can help for common actions such as save, delete or print. For more system-specific actions, one needs to follow broader principles. For example, a button says ‘bold’: does this represent the current *state* of a system or the *action* that will be performed if the button is pressed?

Affordances

These are especially difficult problems in multimedia applications where one may deliberately adopt a non-standard and avant-garde style. How are users supposed to know where to click? The psychological idea of *affordance* says that things may suggest by their shape and other attributes what you can do to them: a handle affords pulling or lifting; a button affords pushing. These affordances can be used when designing novel interaction elements. One can either mimic real-world objects directly, or try to emulate the critical aspects of those objects. What you must not do is depict a real-world object in a context where its normal affordances do not work!

Note also that affordances are not intrinsic, but depend on the background and culture of users. Most computer-literate users will click on an icon. This is not because they go around pushing pictures in art galleries, but because they have learned that this is an affordance of such objects in a computer domain. Similarly, such experienced users may well double click if a single click has no effect, yet novices would not even think of double clicking – after all, double clicking on most real buttons turns them off again!

5.7.3 Appropriate appearance

Presenting information

The way of presenting information on screen depends on the kind of information: text, numbers, maps, tables; on the technology available to present it: character display, line drawing, graphics, virtual reality; and, most important of all, on the purpose for which it is being used. Consider the window in Figure 5.13. The file listing is alphabetic, which is fine if we want to look up the details of a particular file, but makes it very difficult to find recently updated files. Of course, if the list were ordered by date then it would be difficult to find a particular file. Different purposes require different representations. For more complex numerical data, we may be considering scatter graphs, histograms or 3D surfaces; for hierarchical structures, we may consider outlines or organization diagrams. But, no matter how complex the data, the principle of matching presentation to purpose remains.

[[[*** old Figure 3.20 here ****]]]

Figure 5.13 Alphabetic file listing

The issue of presentation has been around for many years, long before computers, interactive systems or HCI! Probably the best source for this issue is Tufte’s book [242]. It is targeted principally at static presentations of information, as in books, but most design principles transfer directly.

However, we have an advantage when presenting information in an interactive system in that it is easy to allow the user to choose among several representations, thus making it possible to achieve different goals. For example, with Macintosh folder windows (as in figure 5.13) the user can click on a column heading and the file list is reordered, so one can look at the files by, say, name or date. This is not an excuse for ignoring the user’s purpose, but means that we can plan for a range of possible uses.

Aesthetics and utility

Remember that a pretty interface is not necessarily a good interface. Ideally, as with any well-designed item, an interface should be aesthetically pleasing. Indeed, good graphic design and attractive displays can increase users’ satisfaction and thus improve productivity.

However, beauty and utility may sometimes be at odds. For example, an industrial control panel will often be built up of the individual controls of several subsystems, some designed by different teams, some bought in. The resulting inconsistency in appearance may look a mess and suggest tidying up. Certainly some of this inconsistency may cause problems. For example, there may be a mix of telephone-style and calculator-style numeric keypads. Under stress it would be easy to mis-key when swapping between these. However, the diversity of controls can also help the operator keep track of which controls refer to which subsystem – any redesign must preserve this advantage.

The conflict between aesthetics and utility can also be seen in many 'well-designed' posters and multimedia systems. In particular, the backdrop behind text must have low contrast in order to leave the text readable; this is often not the case and graphic designers may include excessively complex and strong backgrounds because they look good. The results are impressive, perhaps even award winning, but completely unusable!

On a more positive note, careful application of aesthetic concepts can also aid comprehensibility. An example of this is the idea of the *counter* and use of space that we discussed earlier. In consumer devices these aesthetic considerations may often be the key differentiator between products, for example, the sleek curves of a car. This is not missed by designers of electronic goods: devices are designed to be good to touch and feel as well as look at and this is certainly one of the drivers for the futuristic shapes of the apple iMac family.

[[[Figure of COUNTER near here – see old chapter 3]]]

Making a mess of it: colour and 3D

One of the worst features in many interfaces is the appalling use of colour. This is partly because many monitors only support a limited range of primary colours and partly because, as with the overuse of different fonts in word processors, the designer got carried away. Aside from issues of good taste an overuse of colour can be distracting and, remembering from Chapter 1 that a significant proportion of the population are colour blind, may mean that parts of the text are literally invisible to some users. In general, colour should be used sparingly and not relied upon to give information, but rather to reinforce other attributes.

The increasing use of 3D effects in interfaces has posed a whole new set of problems for text and numerical information. Whilst excellent for presenting physical information and certain sorts of graphs, text presented in perspective can be very difficult to read and the all too common 3D PIE chart is all but useless. We will discuss ways to make 3D actually useful for visualization in Chapter 20!

Design Focus
Checking screen colours

Even non-colour-blind users will find it hard to read text where the intensity of the text and background are similar. A good trick is to adjust the colour balance on your monitor so that it is reduced to greys, or to print screens on a black and white printer. If your screen is unreadable in greyscale then it is probably difficult to read in full colour.

Localization/internationalization

If you travel to different countries, you frequently see a document being word processed, where the text of the document and the file names are in the local language, but all the menus and instructions are still in English. The process of making software suitable for different languages and cultures is called *localization* or *internationalization*.

It is clear that words have to change and many interface construction toolkits make this easy by using *resources*. When the program uses names of menu items, error messages and other text, it does not use the text directly, but instead uses a resource identifier, usually simply a number. A simple database is constructed separately that binds these identifiers to particular words and phrases. A different resource database is constructed for each language, and so the program can be customized to use in a particular country by simply choosing the appropriate resource database.

However, changing the language is only the simplest part of internationalization. Much of the explicit guidance on alignment and layout is dependent on a left to right, top to bottom language such as English and most European languages. This obviously changes completely for other types of language. Furthermore, many icons and images are only meaningful within a restricted cultural context. Despite the apparent international hegemony of Anglo-American culture, one cannot simply assume that its symbols and norms will be universally understood. A good example of this is the use of ticks ✓ and crosses ✗ . In Anglo-American culture these represent opposites, positive and negative, whereas in most of Europe the two are interchangeable.

5.8 Iteration and prototyping

Because human situations are complex and designers are not infallible it is likely that our first design will not be perfect! For this reason almost all interaction design includes some form of iteration of ideas. This often starts early on with paper designs and story boards being demonstrated to colleagues and potential users. Later in the design process one might use mockups of physical devices or tools such as Shockwave or Visual Basic to create prototype versions of software.

Any of these prototypes, whether paper-based or running software, can then be evaluated to see whether they are acceptable and where there is room for improvement. This sort of evaluation, intended to improve designs, is called *formative* evaluation. This is in contrast to *summative* evaluation, which is performed at the end to verify whether the product is good enough. Chapter 9 considers evaluation in detail. One approach is to get an expert to use a set of guidelines, for example the “knowing where you are” list above, and look screen by screen to see of there are any violations. The other main approach is to involve real users either in a controlled experimental setting, or ‘in the wild’ – a real use environment.

The result of evaluating the system will usually be a list of faults or problems and this is followed by a redesign exercise, which is then prototyped, evaluated ... Figure 5.14 shows this process. The end point is when there are no more problems that can economically be fixed.

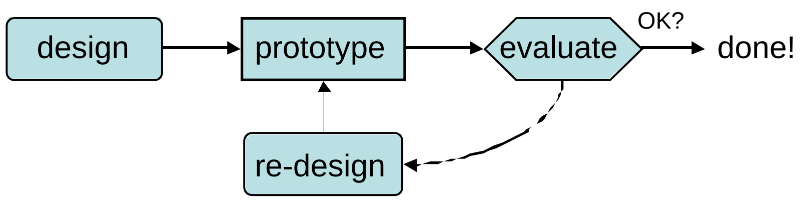


Figure 5.14 Role of prototyping

So iteration and prototyping are the universally accepted 'best practice' approach for interaction design. However, there are some major pitfalls of prototyping, rarely acknowledged in the literature.

Prototyping is an example of what is known as a *hill-climbing* approach. Imagine you are standing somewhere in the open countryside. You walk uphill and keep going uphill as steeply as possible. Eventually you will find yourself at a hill top. This is exactly how iterative prototyping works: you start somewhere, evaluate it to see how to make it better, change it to make it better and then keep in doing this until it can't get any better.

However, hill climbing doesn't always work. Imagine you start somewhere near Cambridge, UK. If you keep moving uphill (and it is very difficult to work out which direction that is because it is very flat!), then eventually you would end up at the top of the Gog Magog hills, the nearest thing around ... all of 300 feet. However, if you started somewhere else you might end up at the top of the Matterhorn. Hill-climbing methods always have the potential to leave you somewhere that is the best in the immediate area, but very poor compared to more distant places. Figure 5.15 shows this schematically: if you start at A you get trapped at the *local maximum* at B, but of you start at C you move up through D to the *global maximum* at E.

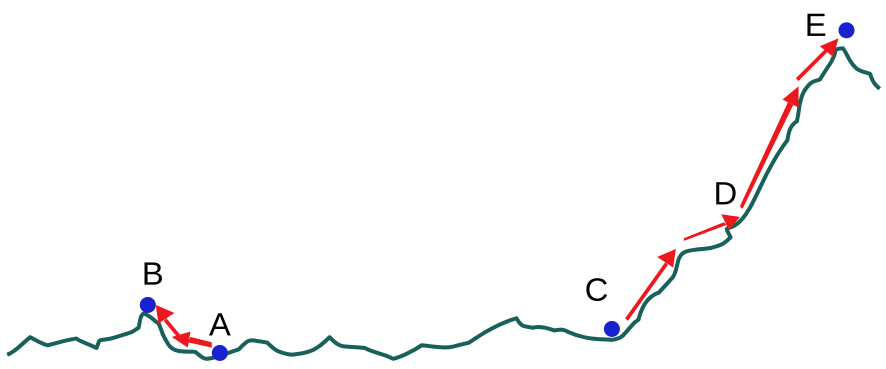


Figure 5.15 Moving little by little … but to where

This problem of getting trapped at local maxima is also possible with interfaces. If you start with a bad design concept you may end at something that is simply a tidied up version of that bad idea!

From this we can see that there are two things you need in order for prototyping methods to work:

- (i) to understand what is wrong and how to improve
- (ii) a good start point

The first is obvious; you cannot iterate the design unless you know what must be done to improve it. The second, however, is needed to avoid local maxima. If you were told that you wanted to climb as high as you could you would probably order a plane to the Himalayas, not Cambridgeshire.

A really good designer might guess a good initial design based on experience and judgment. However, the complexity of interaction design problems means that this insight is hard. Another approach, very common in graphical design, is to have several initial design ideas and drop them one by one as they are developed further. This is a bit like parachuting 10 people at random points of the earth. One of them is perhaps likely to end up near a high mountain.

One of the things that theoretical methods and models, as found in Part III of this book, can do is to help us with both (i) and (ii).

5.9 Summary

We have seen that design in HCI is not just about creating devices or software, but instead is about the whole interaction between people, software, and their environment. Because of this it is good to see the product of design not just as the obvious artefacts but as the whole intervention that changes the existing situation to a new one.

In section 5.2 design was defined as "*achieving goals within constraints*". In the case of interaction design the goals are about improving some aspect of work, home or leisure using technology. The constraints remind us that the final design will inevitably involve *trade-offs* between different design issues and furthermore should never be 'perfect' as cost and timeliness should prevent indefinite tinkering. To achieve good design we must *understand our materials* and in the case of interaction design these materials include not just the computers and technical devices, but also humans. If we treated humans in design with *only* as much care as physical materials it is clear that 'human error' after accidents would be regarded as 'design error' – a good designer understands the natural limitations of ordinary people.

Section 5.3 gave a birds-eye view of the *design process*, which gives a context for much of the rest of this book.

The process starts with understanding the situation as it is and the requirements for change. Section 5.4 provided some simple techniques for dealing with this: getting to know your users, who they are, remembering that they are different from you, but trying to imagine what it is like for them. You can talk to users, but you should also observe them in other ways, as we are all bad at articulating what we do. One way to help retain a user focus in design is to use *personae* – detailed word pictures of imaginary but typical users.

Section 5.5 introduced *scenarios* and rich stories about design, which can help us explore the design space and to discuss potential designs with other designers and potential users. Both scenarios and personae need to be vivid and to include rich contextual details – not just a record of user actions on the system!

The details of potential designs need to be worked out and in section 5.6 we looked at the overall navigation design of the system. We started by looking at local structure, the way one screen, page or state of an application relates to those it immediately links to. The users need to know *where they are*, *what they can do*, *what will happen* when they do things, and *what has happened* in the past. This can aid users as they *goal seek*, move closer towards their goals without having to necessarily understand completely the whole route there. The *global structure* of the application is also important. We saw how *hierarchy diagrams* can give a logical view of an application, which can be used to design menu or site structures. In contrast the user *dialogue* focuses on the flow of user and system actions. One way to do this is using *network diagrams* of screens or states of the system and how they link to one another. Any designed system must also relate to its *environment*: other applications, other web sites, other physical devices.

In section 5.7 we looked at *screen design and layout*. We saw that there were various visual tools that could help us to ensure that the physical structure of our screen emphasized the logical structure of the user interaction. These tools included physical grouping, ordering of items, decoration such as fonts, lines and colour, alignment and the use of white space. These are important both for appropriate display of information and to lay out controls and data entry fields for ease of use. It is also important that controls have appropriate affordances – that is have visual and tactile attributes that suggest their use. Information presented on screen, whether individual items, tabular or graphical, should be appropriate to the user's purpose and this may mean allowing interactions to change the layout, for example re-sort tables by different columns. Aesthetics are also important, but may conflict with utility. Depending on the context you may need to make different trade-offs between these. Good graphical design is an area and a skill all of its own, but some features such as bad use of colour and 3D effects are bad for both aesthetics and usability!

Finally in section 5.8 we saw that iteration is an essential part of virtually any interaction design process because we cannot get things right first time. However iterative methods may get trapped in local maxima. To make iterative processes work we need either extensive personal experience or theoretical understanding to help us get better initial designs.

Exercises

5.1 Use a pocket alarm clock or wristwatch to set yourself alarms every 15 minutes one working day. Write down exactly what you are doing. How surprising is it?

The exercises 5.2, 5.3, 5.4 and 5.5 are based around a nuclear reactor scenario on the web at:: `/e3/scenario/nuclear` You will need to read the scenario in order to answer these exercises.

5.2 Comment on the user of colour in the Alarm Control, Emergency Shutdown and Emergency Confirm panels (figure CS.2)

5.3 Comment on the use of layout and other elements in the control panels (figures CS.1, CS.2 and CS.3), including the way in which various visual elements support or hinder logical grouping and sequence

5.4 Working through the accident scenario explain why the various problems arise

5.5 Suggest potential ways of improving the interface to avoid a similar problem recurring

Recommended reading

- Preece, J., Rogers, Y. & Sharp, H. (2002) *Interaction Design: Beyond Human-Computer Interaction*. New York, NY: John Wiley & Sons
  A general text book in interaction design with especially strong focus on evaluation.

- J. Carroll (editor). Special Issue on "Scenario-Based System Development". Interacting with Computers 13(1), 2000.
  Contributions from several authors on using scenarios in design.
- J. Carroll (2000). Making Use：Scenario-Based Design of Human-Computer Interactions. MIT Press.
  John Carroll’s own book dedicated solely to using scenarios in design.
- McGrenere, J., and Ho, W. (2000). Affordances: Clarifying and evolving a concept. Proceedings of Graphics Interface 2000, 179-186
  This paper reviews all the major work on affordances from Gibson’s original definition and focuses especially on Norman’s popularization of the word which has been the way many encounter it. She also reviews work of Bill Gaver, who is probably the first person to use affordance as a concept within HCI.
- E. Tufte. *Envisioning Information*. Cheshire, US, Graphics Press, 1990.
  and
  E. Tufte, *The Visual Display of Quantitative Information*, Cheshire, US Graphics Press, 1997.
  Tufte's books are the 'must read' for graphical presentation, packed with examples and pictures from timetables to Napoleon's disastrous Russian campaign.
-

References

[[[all new]]]

[[LC98]] K. Larson and M Czerwinski. Web Page Design: Implications of Memory, Structure and Scent for Information Retrieval, In Proceedings of CHI 98, Human Factors in Computing Systems (LA, April 21–23, 1998), ACM Press, 25–32.

[[GDP99]] Bill Gaver, Tony Dunne and Elena Pacenti (1999). Design: Cultural probes. Interactions, ACM Press, 6(1):21–29.

[[HCRCR02]] Hemmings, T., Crabtree, A., Rodden, T., Clarke, K., and Rouncefield, M. (2002) “Domestic probes and the design process”, Proceedings of the 11th European Conference on Cognitive Ergonomics, pp. 187-193, Catania, Italy: European Association of Cognitive Ergonomics.

[[H71]] Hansen, W. J. User engineering principles for interactive systems. In AFIPS Conference Proceedings 39, AFIPS Press, 1971, pp. 523--532.
reprinted in: W.J. Hansen. User engineering principles for interactive systems. In Interactive Programming Environments, D.R. Barstow, H.E. Shrobe, and E. Sandewall, editors, McGraw-Hill, New York, 1984, pp. 217--231