Summary(3rd Chapter)

3.2 MODELS OF INTERACTION

Interaction involves at least two participants:
the user and the system. Both are complex, as we have seen, and are very
different from each other in the way that they communicate and view the domain
and the task. The interface must therefore effectively translate between them to allow
the interaction to be successful. This translation can fail at a number of points and
for a number of reasons. The use of models of interaction can help us to understand
exactly what is going on in the interaction and identify the likely root of difficulties.

3.2.1 The terms of interaction

Traditionally, the purpose of an interactive system is to aid a user in accomplishing goals
from some application domain.**A domain** defines an area of expertise and knowledge in
some real-world activity. Some examples of domains are graphic design, authoring and
process control in a factory.**Tasks** are operations to manipulate the concepts of a
domain. **A goa**l is the desired output from a performed task.An **intention** is a specific
action required to meet the goal.
Task analysis involves the identification of the problem space for the user of an interactive
system in terms of the domain, goals,
intentions and tasks.
The System and User are each described by means
of a language that can express concepts relevant in the domain of the application.
The System’s language we will refer to as the core language and the User’s language
we will refer to as the task language. The core language describes computational
attributes of the domain relevant to the System state, whereas the task language
describes psychological attributes of the domain relevant to the User state.

3.2.2 The execution–evaluation cycle

The interactive cycle can be divided into two major phases: execution and evaluation. These can then be subdivided into further stages, seven in all. The stages in Norman's model of interaction are as follows:

1. Establishing the goal.
2. Forming the intention.
3. Specifying the action sequence.
4. Executing the action.
5. Perceiving the system state.
6. Interpreting the system state.
7. Evaluating the system state with respect to the goals and intentions.

Example:

Norman uses a simple example of switching on a light to illustrate this cycle. Imagine you are sitting reading as evening falls. You decide you need more light; that is you establish the goal to get more light. From there you form an intention to switch on the desk lamp, and you specify the actions required, to reach over and press the lamp switch. If someone else is closer the intention may be different – you may ask them to switch on the light for you. Your goal is the same but the intention and actions are different. When you have executed the action you perceive the result, either the light is on or it isn't and you interpret this, based on your knowledge of the world. For example, if the light does not come on you may interpret this as indicating the bulb has blown or the lamp is not plugged into the mains, and you will formulate new goals to deal with this. If the light does come on, you will evaluate the new state according to the original goals – is there now enough light? If so, the cycle is complete. If not, you may formulate a new intention to switch on the main ceiling light as well.

Norman uses this model of interaction to demonstrate why some interfaces cause problems to their users. He describes these in terms of the gulfs of execution and the gulfs of evaluation. As we noted earlier, the user and the system do not use the same terms to describe the domain and goals – remember that we called the language of the system the core language and the language of the user the task language. The gulf of execution is the difference between the user's formulation of the actions to reach the goal and the actions allowed by the system. If the actions allowed by the system correspond to those intended by the user, the interaction will be effective. The interface should therefore aim to reduce this gulf.

The gulf of evaluation is the distance between the physical presentation of the system state and the expectation of the user. If the user can readily evaluate the presentation in terms of his goal, the gulf of evaluation is small. The more effort that is required on the part of the user to interpret the presentation, the less effective the interaction.

Human error – slips and mistakes

Human errors are often classified into slips and mistakes. We can distinguish these using Norman's gulf of execution.

If you understand a system well you may know exactly what to do to satisfy your goals – you have formulated the correct action. However, perhaps you mistype or you accidentally press the mouse button at the wrong time. These are called slips; you have formulated the right action, but fail to execute that action correctly.

However, if you don't know the system well you may not even formulate the right goal. For example,

you may think that the magnifying glass icon is the 'find' function, but in fact it is to magnify the text. This is called a mistake.

.

3.2.3 The interaction framework

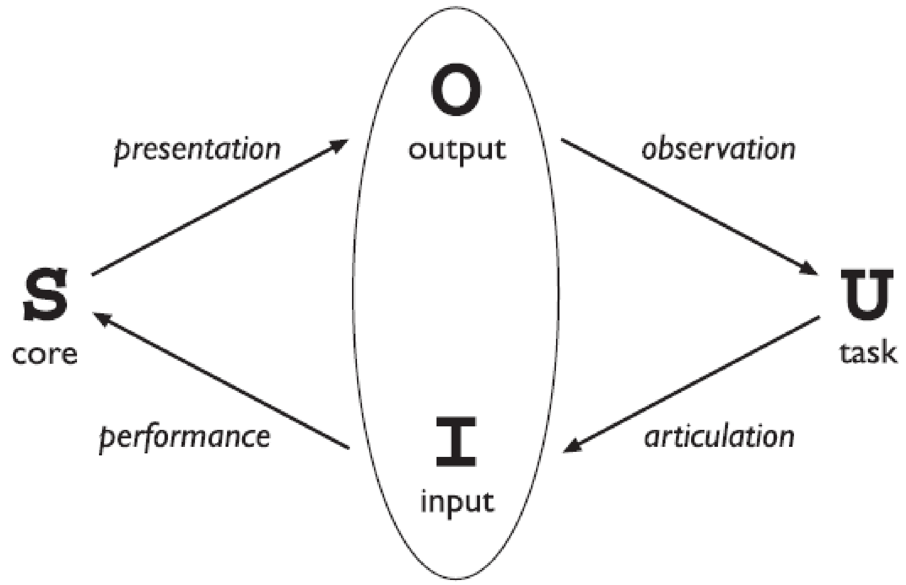


Figure 3.2 Translations between components

The nodes represent the four major components in an interactive system
– the System, the User, the Input and the Output. Each component has its own
language.Input and Output together form the Interface.
The User begins the interactive
cycle with the formulation of a goal and a task to achieve that goal. The only way
the user can manipulate the machine is through the Input, and so the task must be
articulated within the input language. The input language is translated into the core
language as operations to be performed by the System. The System then transforms
itself as described by the operations; the execution phase of the cycle is complete and
the evaluation phase now begins. The System is in a new state, which must now
be communicated to the User. The current values of system attributes are rendered
as concepts or features of the Output. It is then up to the User to observe the Output
and assess the results of the interaction relative to the original goal, ending the evaluation
phase and, hence, the interactive cycle. There are four main translations
involved in the interaction: articulation, performance, presentation and observation.

3.4 ERGONOMICS

Ergonomics (or human factors) is traditionally the study of the physical characteristics
of the interaction: how the controls are designed, the physical environment in
which the interaction takes place, and the layout and physical qualities of the screen.

3.4.1 Arrangement of controls and displays

The exact organization that this will
suggest will depend on the domain and the application, but possible organizations
include the following:
functional controls and displays are organized so that those that are functionally
related are placed together;
sequential controls and displays are organized to reflect the order of their use in a
typical interaction (this may be especially appropriate in domains where a particular
task sequence is enforced, such as aviation);
frequency controls and displays are organized according to how frequently they are
used, with the most commonly used controls being the most easily accessible.

3.4.2 The physical environment of the interaction

The first consideration here is the size of the users. Obviously this is going to vary considerably. However, in any system the smallest user should be able to reach all the controls (this may include a user in a wheelchair), and the largest user should not be cramped in the environment.

3.4.3 Health issues

Physical position: As we noted in the previous section, users should be able to reach all controls comfortably and see all displays. Users should not be expected to stand for long periods and, if sitting, should be provided with back support.

Temperature: Although most users can adapt to slight changes in temperature without adverse effect, extremes of hot or cold will affect performance and, in excessive cases, health.

Lighting: The lighting level will again depend on the work environment. However, adequate lighting should be provided to allow users to see the computer screen without discomfort or eyestrain.

Noise: Excessive noise can be harmful to health, causing the user pain, and in acute cases, loss of hearing.

Time: The time users spend using the system should also be controlled.

3.4.4 The use of color

Colors used in the display should be as distinct as possible and the distinction should not be affected by changes in contrast. Blue should not be used to display critical information. If color is used as an indicator it should not be the only cue: additional coding information should be included.
The colors used should also correspond to common conventions and user expectations. Red, green and yellow are colors frequently associated with stop, go and standby respectively. Therefore, red may be used to indicate emergency and alarms; green, normal activity; and yellow, standby and auxiliary function. These conventions should not be violated without very good cause.

However, we should remember that color conventions are culturally determined.
For example, red is associated with danger and warnings in most western cultures, but in China it symbolizes happiness and good fortune. The color of mourning is black in some cultures and white in others. Awareness of the cultural associations of color is particularly important in designing systems and websites for a global market.

3.5 INTERACTION STYLES

Here we introduce the most common interface
styles and note the different effects these have on the interaction. There are a number of common interface styles including

1. command line interface
2. menus
3. natural language
4. question/answer and query dialog
5. form-fills and spreadsheets
6. WIMP
7. point and click
8. three-dimensional interfaces.

.

3.5.1 Command line interface

Command line interfaces are powerful in that they offer direct access to system functionality (as opposed to the hierarchical nature of menus), and can be combined to apply a number of tools to the same data. They are also flexible:
However, this flexibility and power brings with it difficulty in use and learning. Commands must be remembered, as no cue is provided in the command line to indicate which command is needed. They are therefore better for expert users than for novices. This problem can be alleviated a little by using consistent and meaningful commands and abbreviations. The commands used should be terms within the vocabulary of the user rather than the technician.

3.5.2 Menus

In a menu-driven interface, the set of options available to the user is displayed on the screen, and selected using the mouse, or numeric or alphabetic keys. Since the options are visible they are less demanding of the user, relying on recognition rather than recall.This is a restricted form of a full WIMP system

3.5.3 Natural language

Perhaps the most attractive means of communicating with computers, at least at first glance, is by natural language..Unfortunately the ambiguity of natural language makes it very difficult for a machine to understand.For example, the word 'pitch' may refer to a sports field, a throw, a waterproofing substance or even, colloquially, a territory. We often rely on the context and our general knowledge to sort out these ambiguities.
However, systems can be built to understand restricted subsets of a language. For a known and constrained domain, the system
can be provided with sufficient information to disambiguate terms.The use of natural language in restricted domains is relatively successful, but it is debatable whether this can really be called natural language.

3.5.4 Question/answer and query dialog

Question and answer dialog is a simple mechanism for providing input to an application in a specific domain. The user is asked a series of questions (mainly with yes/no responses, multiple choice, or codes) and so is led through the interaction step by step. An example of this would be web questionnaires.These interfaces are easy to learn and use, but are limited in functionality and
power.

3.5.5 Form-fills and spreadsheets

Form-filling interfaces are used primarily for data entry but can also be useful in data retrieval applications.
Spreadsheets are a sophisticated variation of form filling.

3.5.6 The WIMP interface

WIMP stands for windows, icons, menus and pointers (sometimes windows, icons, mice and pull-down menus), and is the default interface style for the majority of interactive computer systems in use today, especially in the PC and desktop workstation arena.

3.5.7 Point-and-click interfaces

In most multimedia systems and in web browsers, virtually all actions take only a single click of the mouse button.
This point-and-click interface style is obviously closely related to the WIMP style.
The point-and-click style has been popularized by world wide web pages, which incorporate all the above types of point-and-click navigation: highlighted words, maps and iconic buttons

3.5.8 Three-dimensional interfaces

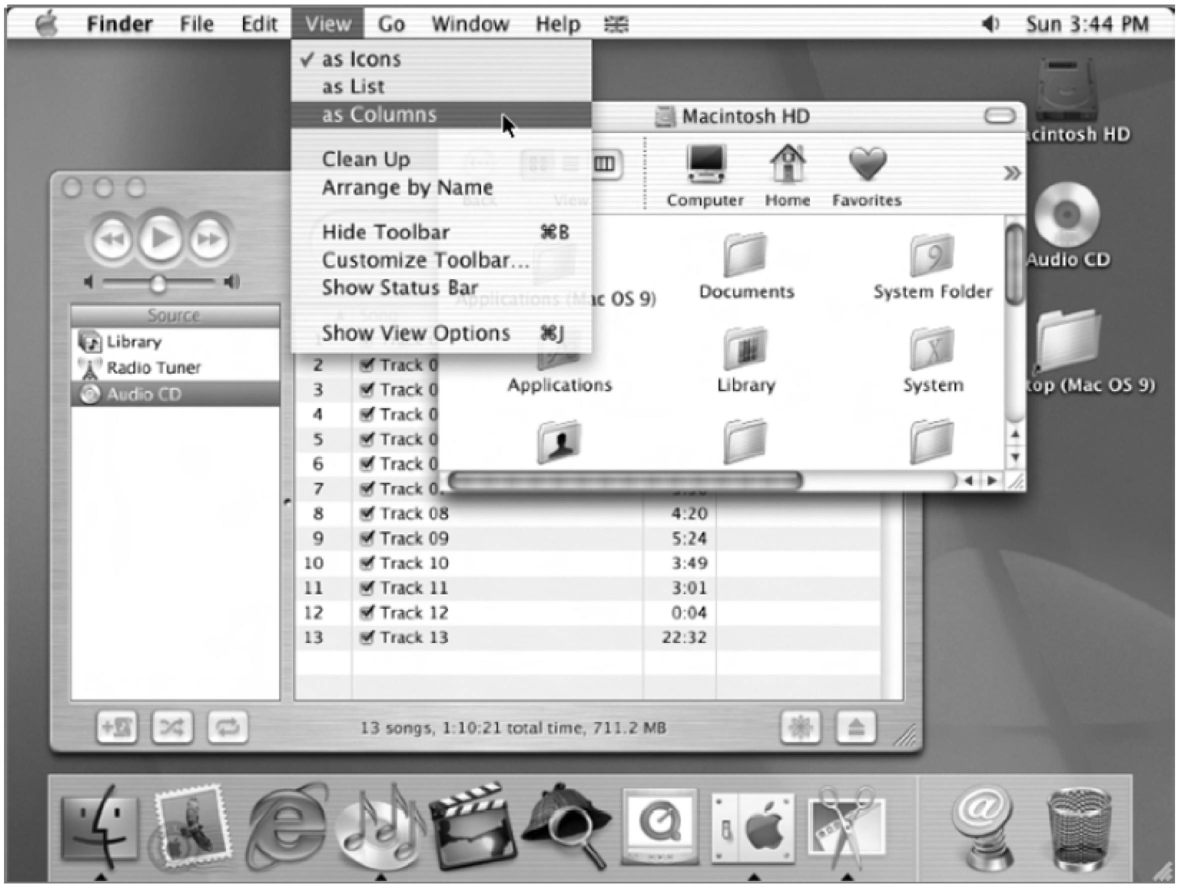
There is an increasing use of three-dimensional effects in user interfaces. The most obvious example is virtual reality

3.6 ELEMENTS OF THE WIMP INTERFACE

We have already noted the four key features of the WIMP interface that give it its name – windows, icons, pointers and menus – and we will now describe these in turn.

3.6.1 Windows

Windows are areas of the screen that behave as if they were independent terminals in their own right. A window can usually contain text or graphics, and can be moved.



3.6.2 Icons

A small picture is used to represent a closed window, and this representation is known as an icon. By allowing icons, many windows can be available on the screen at the same time, ready to be expanded to their full size by clicking on the icon. Shrinking a window to its icon is known as iconifying the window.

3.6.4 Menus

A menu presents a choice of operations or services that can be performed by the system at a given time.Menus are inefficient when they have too many items, and so cascading menus are utilized, in which item selection opens up another menu adjacent

to the item, allowing refinement of the selection. Several layers of cascading
menus can be used.

3.6.5 Buttons

Buttons are individual and isolated regions within a display that can be selected
by the user to invoke specific operations.
toggle buttons can be grouped together to allow a user to select one feature from a
set of mutually exclusive options, such as the size in points of the current font. These
are called radio buttons, since the collection functions much like the old-fashioned
mechanical control buttons on car radios

3.6.6 Toolbars

The function of this
toolbar is similar to a menu bar, but as the icons are smaller than the equivalent
text more functions can be simultaneously displayed.

3.7 INTERACTIVITY

.It is worth remembering that interactivity is the defining feature of an interactive
system. This can be seen in many areas of HCI. For example, the recognition rate
for speech recognition is too low to allow transcription from tape, but in an airline
reservation system, so long as the system can reliably recognize yes and no it can
reflect back its understanding of what you said and seek confirmation. Speech-based
input is difficult, speech-based interaction easier.
In older computer systems, the order of interaction was largely determined by the
machine. You did things when the computer was ready. In WIMP environments, the
user takes the initiative, with many options and often many applications simultaneously
available. The exceptions to this are pre-emptive parts of the interface, where
the system for various reasons wrests the initiative away from the user, perhaps
because of a problem or because it needs information in order to continue.
The major example of this is modal dialog boxes. It is often the case that when a
dialog box appears the application will not allow you to do anything else until the
dialog box has been completed or cancelled. In some cases this may simply block the
application, but you can perform tasks in other applications. In other cases you can
do nothing at all until the dialog box has been completed. An especially annoying

example is when the dialog box asks a question, perhaps simply for confirmation of
an action, but the information you need to answer is hidden by the dialog box!

3.8 THE CONTEXT OF THE INTERACTION

The user may also lose motivation if a system is introduced that does not match
the actual requirements of the job to be done. Often systems are chosen and introduced
by managers rather than the users themselves. In some cases the manager's
perception of the job may be based upon observation of results and not on actual
activity. The system introduced may therefore impose a way of working that is
unsatisfactory
to the users. If this happens there may be three results: the system will be
rejected, the users will be resentful and unmotivated, or the user will adapt the
intended interaction to his own requirements. This indicates the importance of
involving actual users in the design process.
On the other hand, the introduction of new technology may prove to be a motivation
to users, particularly if it is well designed, integrated with the user's current
work, and challenging. Providing adequate feedback is an important source of motivation
for users. If no feedback is given during a session, the user may become bored,
unmotivated or, worse, unsure of whether the actions performed have been successful.
In general, an action should have an obvious effect to prevent this confusion and
to allow early recovery in the case of error.

3.9 EXPERIENCE, ENGAGEMENT AND FUN

Ask many in HCI about usability and they may use the words 'effective' and
'efficient'. Some may add 'satisfaction' as well. This view of usability seems to stem
mainly from the Taylorist tradition of time and motion studies: if you can get the
worker to pull the levers and turn the knobs in the right order then you can shave
10% off production costs.
However, users no longer see themselves as cogs in a machine. Increasingly,
applications are focussed outside the closed work environment: on the home, leisure,
entertainment, shopping. It is not sufficient that people can use a system, they must
want to use it.
Even from a pure economic standpoint, your employees are likely to work better
and more effectively if they enjoy what they are doing!

3.9.1 Understanding experience

Shopping is an interesting example to consider. Most internet stores allow you to buy things, but do you go shopping? Shopping is as much about going to the shops, feeling the clothes, being with friends. You can go shopping and never intend to spend money. Shopping is not about an efficient financial transaction, it is an experience.

But experience is a difficult thing to pin down; we understand the idea of a good experience, but how do we define it and even more difficult how do we design it?

3.9.2 Designing experience

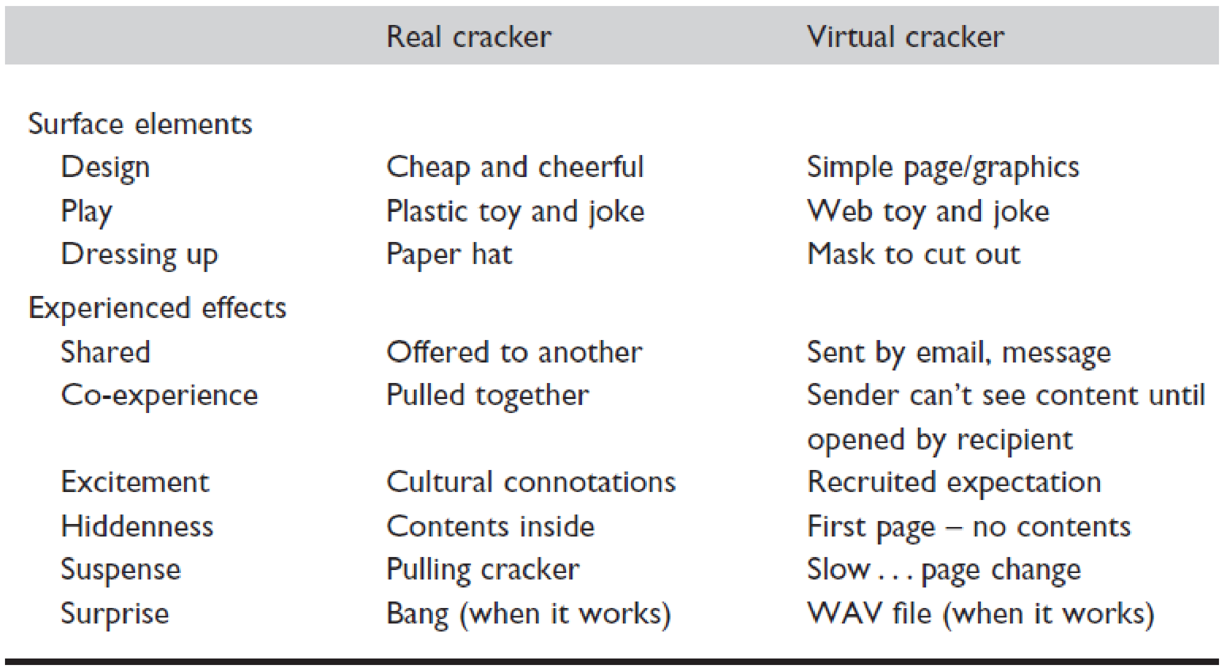
Some of the authors were involved in the design of virtual Christmas crackers. These are rather like electronic greetings cards, but are based on crackers. For those who have not come across them, Christmas crackers are small tubes of paper between 8 and 12 inches long (20–30 cm). Inside there are a small toy, a joke or motto and a paper hat. A small strip of card is threaded through, partly coated with gunpowder. When two people at a party pull the cracker, it bursts apart with a small bang from the gunpowder and the contents spill out.

The virtual cracker does not attempt to fully replicate each aspect of the physical characteristics and process of pulling the cracker, but instead seeks to reproduce the experience. To do this the original crackers experience was deconstructed and each aspect of the experience produced in a similar, but sometimes different, way in the new media. Table 3.1 shows the aspects of the experience deconstructed and reconstructed in the virtual cracker.

For example, the cracker contents are hidden inside; no one knows what toy or joke will be inside. Similarly, when you create a virtual cracker you normally cannot see the contents until the recipient has opened it. Even the recipient initially sees a page with just an image of the cracker; it is only after the recipient has clicked on the 'open' icon that the cracker slowly opens and you get to see the joke, web toy and mask.

The mask is also worth looking at. The first potential design was to have a picture of a face with a hat on it – well, it wouldn't rank highly on excitement! The essential feature of the paper hat is that you can dress up. An iconic hat hardly does that. Instead the cracker has a link to a web page with a picture of a mask that you can print, cut out and wear. Even if you don't actually print it out, the fact that you could changes the experience – it is some dressing up you just happen not to have done yet.

Table 3.1 The crackers experience [101]



	Real cracker	Virtual cracker
Surface elements		
Design	Cheap and cheerful	Simple page/graphics
Play	Plastic toy and joke	Web toy and joke
Dressing up	Paper hat	Mask to cut out
Experienced effects		
Shared	Offered to another	Sent by email, message
Co-experience	Pulled together	Sender can't see content until opened by recipient
Excitement	Cultural connotations	Recruited expectation
Hiddenness	Contents inside	First page – no contents
Suspense	Pulling cracker	Slow . . . page change
Surprise	Bang (when it works)	WAV file (when it works)

3.9.3 Physical design and engagement

Designers are faced with many constraints:
Ergonomic: You cannot physically push buttons if they are too small or too close.
Physical: The size or nature of the device may force certain positions or styles of control, for example, a dial like the one on the washing machine would not fit on the MiniDisc controller; high-voltage switches cannot be as small as low-voltage ones.
Legal and safety: Cooker controls must be far enough from the pans that you do not burn yourself, but also high enough to prevent small children turning them on.
Context and environment: The microwave's controls are smooth to make them easy to clean in the kitchen.
Aesthetic: The controls must look good.
Economic: It must not cost too much!
These constraints are themselves often contradictory and require trade-offs to be made. For example, even within the safety category front-mounted controls are better in that they can be turned on or off without putting your hands over the pans and hot steam, but back-mounted controls are further from children's grasp. The MiniDisc player is another example; it physically needs to be small, but this means there is not room for all the controls you want given the minimum size that can

be manipulated. In the case of the cooker there is no obvious best solution and so different designs favor one or the other. In the case of the MiniDisc player the end knob is multi-function. This means the knob is ergonomically big enough to turn and physically small enough to fit, but at the cost of a more complex interaction style.

Fluidity: The extent to which the physical structure and manipulation of the device naturally relate to the logical functions it supports.
As well as being fluid in action, some controls portray by their physical appearance the underlying state they control. For example, the dial on the washing machine both sets the program and reflects the current stage in the washing cycle as it turns. A simple on/off switch also does this. However, it is also common to see the power on computers and hifi devices controlled by a push button – press for on, then press again for off. The button does not reflect the state at all. When the screen is on this is not a problem as the fact that there is something on the screen acts as a very immediate indicator of the state. But if the screen has a power save then you might accidentally turn the machine off thinking that you are turning it on! For this reason, this type of power button often has a light beside it to show you the power is on.
A simple switch tells you that itself !

3.9.4 Managing value

If we want people to want to use a device or application we need to understand their personal values. Why should they want to use it? What value do they get from using it? Now when we say value here we don't mean monetary value, although that may be part of the story, but all the things that drive a person.
This means that not only must we understand people's value systems, but we must be able to offer gains sooner as well as later, or at least produce a very good demonstration of potential future gains so that they have a perceived current value.

Summary

4.1 Intro

The designer of an interactive system is posed with **two open questions**:

1. How can an interactive system be developed to ensure its usability?
2. How can the usability of an interactive system be demonstrated or measured?

One approach to answering these questions is by **means of example**, in which successful interactive systems are commonly believed to enhance usability and, therefore, serve as paradigms for the development of future products. Though it is sometimes hard to find a consensus for the reason behind the success.

4.2.1-4.2.15 - Design Paradigms for Interaction

4.2.1 Time Sharing

By the 1960s it was becoming apparent that the explosion of growth in computing power would be wasted if there was not an equivalent explosion of ideas about how to channel that power.

Time sharing: The concept of a single computer supporting multiple users.

Previously, computing was restricted to performing batch jobs (not-interactive). Time sharing systems of the 1960 **made programming a truly interactive venture**. The computer could now project itself as a dedicated partner with each individual user and the increased throughput of information between user and computer allowed the human to become a more reactive and spontaneous collaborator. Real human-computer-interaction was now possible.

4.2.2 Video Display Units

As early as the mid-1950s researchers were experimenting with the possibility of presenting and manipulating information from a computer in the form of images on a **video display unit (VDU)**. These display screens could provide a more suitable medium than a paper printout for presenting vast quantities of information.

Sketchpad: Program by MIT grad (I. Sutherland) that first realized the capabilities of visual images on a computer. It allowed a computer operator to use the computer to create sophisticated visual models on a display screen and storing and retrieving them in/from the computer just like any other data.
It was a kind of simulation language that enabled **computers to translate abstractions into perceptually concrete forms**. It was a model for totally new ways of operating computers i.e by changing something on the display screen, it was possible (via Sketchpad)

to change something in the computer's memory.

Sketchpad demonstrated **two** important ideas:
i) Computers could be used for more than just data processing.
ii) How important the contribution of one creative mind (I. Sutherland) coupled with a dogged
determination to see the idea through could be to the entire history of computing.

4.2.3 Programming Toolkits

Douglas Engelbart's ambition was to use computer technology as a means of complementing human problem-solving activity. Providing the right **toolkit** produces computing equipment that aids human problem solving ability.

Augmenting Man's Intellect: Increasing capability to approach a complex problem situation, gain comprehension to suit particular needs and to derive solutions.

Bootstrapping: The idea of building components of a computer system that will allow us to rebuild a more complex system. It has been used to a great extent in all of computing.

Power of programming toolkits:
→ Small, well-understood components can be composed in fixed ways in order to create larger tools.
→ Once these larger tools become understood, they can continue to be composed with other tools and the process continues.

4.2.4 Personal Computing

According to Seymour Papert, no matter how powerful a system may be, it will always be more powerful if it's easier to use.

LOGO: Graphics programming language for children. A computer controlled mechanical turtle that dragged a pen along a surface to trace its path.

Following this, Alan Kay's view of the future of computing was embodied in small, powerful machines which were dedicated to single users, that is personal computers. Now, it's difficult to distinguish between a personal computer and a mainframe.

Dynabook: Kay's vision of the ultimate handheld pc. (Something like ipads/ surface books we have today)

4.2.5 Window systems and the WIMP interface

Humans are able to think about more than one thing at a time, and in accomplishing some piece of work, they frequently interrupt their current train of thought to pursue some other related piece of work. Not allowing these diversions does not correspond to that standard working pattern of humans. **Personal computers should be flexible in their ability to 'change the topic' as humans have.**

Each interaction between a human and computer is like a dialogue between two partners. It is therefore necessary for the computer dialog partner to present the context of each thread of dialog so that the user can distinguish them. One presentation mechanism for achieving this dialog partitioning is to physically separate the presentation of the different logical threads of the user–computer conversation on the display device. The **window** is the common mechanism associated with these physically and logically separate display spaces.

WIMP Interface: Interaction / interfaces based on windows, icons, menus and pointers

4.2.6 Metaphor

Metaphors are used quite successfully to teach new concepts in terms of ones which are already understood. Metaphors are used to describe the functionality of many interaction widgets, such as windows, menus, buttons and palettes.

Xerox Alto and **Star:** First workstations based on the **metaphor of the office desktop**. The majority of the management tasks on a standard workstation have to do with file manipulaiton. Linking the set of tasks associated with file manipulation to the filing tasks in a typical office environment makes the actual computerized tasks easier to understand.

Other Examples: Spreadsheet metaphor for accounting and financial modelling.

Question 1: What are the dangers/problems of using the metaphor paradigm ?

The danger of a metaphor is usually realized after the initial honeymoon period.

Metaphors can sometimes get in the way of the user understanding the

computer:
When **word processors** were first introduced, they relied heavily on the typewriter metaphor. The keyboard of a computer closely resembles that of a standard typewriter, so it seems like a good metaphor from which to start. However, the behavior of a word processor is different from any typewriter. **e.g:** the space key on a typewriter is passive, producing nothing on the piece of paper and just moving the guide further along the current line. For a typewriter, a space is not a character. However, for a word processor, the blank space is a character which must be inserted within a text just as any other character is inserted. So an experienced typist is not going to be able to correctly predict the behavior of pressing the spacebar on the keyboard by appealing to his experience with a typewriter. Whereas the typewriter metaphor is beneficial for providing a preliminary understanding of a word processor, the analogy is inadequate for promoting a full understanding of how the word processor works. Another **e.g:** Although the desktop metaphor is initially appealing, it falls short in the computing world because there are no office equivalents for ejecting a floppy disk or printing a document.

Metaphors can sometimes portray cultural bias:
With the growing internationalization of software, it should not be assumed that a metaphor
will apply across national boundaries. A meaningless metaphor will only add another layer of complexity between the user and the system.

An extreme example of Metaphor:
In a VR system, the metaphor is not simply captured on a display screen. Rather, the user is also portrayed within the metaphor, literally creating an alternative, or virtual, reality. Any actions that the user performs are supposed to become more natural and so more movements of the user are interpreted, instead of just keypresses, button clicks and movements of an external pointing device. A VR system also needs to know the location and orientation of the user. Consequently, the user is often 'rigged' with special tracking devices so that the system can locate them and interpret their motion correctly.

Question 2: What features of a modern word processor break the metaphor of composition with pen (or typewriter) and paper?

→ The functionality of the space key in typewriting versus word processing. For a typewriter, the space key is passive; it merely moves the insertion point one space to the right. In a word processor, the space key is active, as it inserts a character (the space character) into the document. The functionality of the typewriter space key is produced by the movement keys for the word processor (typically an arrow key pointing right to move forward within one line). In fact, much of the functionality that we have come to expect of a word processor is radically different from that expected of a typewriter, so much so that the typewriter as a metaphor for word processing is not all that instructive. In practice, modern typewriters have begun to borrow from word processors when defining their functionality!

4.2.7 Direct Manipulation / Action Paradigm

In the early 1980s, designers were beginning to see that their products were gaining popu-
larity as their visual content increased. In a standard command line interface, the only way to get any feedback on the results of previous interaction is to **know that you have to ask for it** and to **know how to ask for it.**

Rapid Feedback (Rapid visual and audio feedback on a high-resolution display screen or through a high-quality sound system makes it possible to provide evaluative information for every executed user action) is just one feature of the interaction technique known as direct manipulation.

Ben Shneiderman highlights the following **features of a direct manipulation interface**:
→ Visibility of the objects of interest
→ Incremental action at the interface with **rapid feedback** on all actions
→ Reversibility of all actions, so that users are encouraged to explore without severe penalties
→ Syntactic correctness of all actions, so that every user action is a legal operation
→ Replacement of complex command languages with actions to directly manipulate the visible objects (and, hence, the name direct manipulation).

e.g: The direct manipulation interface for the desktop metaphor requires that the documents and folders are made visible to the user as icons which represent the underlying files and directories. Operations like Moving a file from one folder to another, work as good metaphors for moving real world files from one place to another. But, if these operations have to be performed through the terminal like before, there remains a chance of syntactically incorrect commands whereas, **it is impossible to formulate a syntactically incorrect move command with the pick-up-and-drag style direct manipulation of UI widgets.**

→ **Model-World Metaphor:** In a system built on the model-world metaphor, the interface is itself a world where the user can act, and which changes state in response to user actions. The world of interest is explicitly represented and there is no intermediary between user and world. Appropriate use of the model-world metaphor can create the sensation in the user of acting upon the objects of the task domain themselves. We call this aspect of directness direct engagement.

→ **A consequence of the direct manipulation paradigm** is that there is no longer a clear distinction between input and output. The output expressions are used to formulate subsequent input expressions. The document icon is an output expression in the desktop metaphor, but that icon is used by the user to articulate the move operation.

→ Somewhat related to the **visualization provided by direct manipulation is the WYSIWYG paradigm**, which stands for 'what you see is what you get'.

Discuss the ways in which a full-page word processor is or is not a direct manipulation interface for editing a document using Shneiderman's criteria.

Answer:

→ We will answer the first point by evaluating the word processor relative to the criteria for direct manipulation given by Shneiderman.

Visibility of the objects of interest

The most important objects of interest in a word processor are the words themselves. Indeed, the visibility of the text on a continual basis was one of the major usability advances in moving from line-oriented to display-oriented editors. Depending on the user's application, there may be other objects of interest in word processing that may or may not be visible. For example, are the margins for the text on screen similar to the ones which would eventually be printed? Is the spacing within a line and the line breaks similar? Are the different fonts and formatting characteristics of the text visible (without altering the spacing)? Expressed in this way, we can see the visibility criterion for direct manipulation as very similar to the criteria for a WYSIWYG interface.

Incremental action at the interface with rapid feedback on all actions

We expect from a word processor that characters appear in the text as we type them in at the keyboard, with little delay. If we are inserting text on a page, we might also expect that the format of the page will adjust immediately to accommodate the new changes. Various word processors do this reformatting immediately, whereas with others changes in page breaks may take some time to be reflected. One of the other important actions which requires incremental and rapid feedback is movement of the window using the scroll button. If there is a significant delay between the input command to move the window down and the actual movement of the window on screen, it is quite possible that the user will 'overshoot' the target when using the scrollbar button.

Reversibility of all actions, so that users are encouraged to explore without severe penalties
Single-step undo commands in most word processors allow the user to recover from the last action performed. One problem with this is that the user must recognize the error before doing any other action. More sophisticated undo facilities allow the user to retrace back more than one command at a time. The kind of exploration this reversibility provides in a word processor is best evidenced with the ease of experimentation that is now available for formatting changes in a document (font types and sizes and margin changes). One problem with the ease of exploration is that emphasis may move to the look of a document rather than what the text actually says (style over content).
Syntactic correctness of all actions, so that every user action is a legal operation
WYSIWYG word processors usually provide menus and buttons which the user uses to articulate many commands. These interaction mechanisms serve to constrain the input language to allow only legal input from the user. Document markup systems, such as HTML and LaTeX, force the user to insert textual commands (which may be erroneously entered by the user) to achieve desired formatting effects.

Replacement of complex command languages with actions to manipulate directly the visible objects
The case for word processors is similar to that described above for syntactic correctness. In addition, operations on portions of text are achieved many times by allowing the user to highlight the text directly with a mouse (or arrow keys). Subsequent action on that text, such as moving it or copying it to somewhere else, can then be achieved more directly by allowing the user to 'drag' the selected text via the mouse to its new location.

4.2.8 Language Paradigm vs Action Paradigm

Whereas it is true that direct manipulation interfaces make some tasks easier to perform correctly, it is equally true that some tasks are more difficult, if not impossible. Contrary to popular wisdom, **it is not generally true that actions speak louder than words.** Direct manipulation was of the interface as a **replacement** for the underlying system as the world of interest to the user whereas, **The Language Paradigm** represents the interface as the **interlocutor or mediator** between the user and the system.
Language Paradigm: The user-system communication is by means of indirect language

instead of direct
Actions.
Two meaningful interpretations of Language Paradigm:
1) The first requires that the user understands how the underlying system functions and the interface as interlocutor need not perform much translation. In fact, this interpretation of the language paradigm is similar to the kind of interaction which existed before direct manipulation interfaces were around.

2) The second interpretation does not require the user to understand the underlying system's structure. The interface serves a more active role, as it must interpret between the intended operation as requested by the user and the possible system operations that must be invoked to satisfy that intent. Because it is more active, some people refer to the interface as an agent in these circumstances. **e.g:** We can see this kind of language paradigm at work in an **information retrieval system**. You may know what kind of information is in some internal system database, such as the UK highway code, but you would not know how that information is organized. If you had a question about speed limits on various roads, how would you ask? The answer in this case is that you would ask the question in whatever way it comes to mind, typing in a question such as, 'What are the speed limits on different roads?' You then leave it up to the interface agent to reinterpret your request as a legal query to the highway code database.

Compare Action Paradigm and Language Paradigm.

Whatever interpretation we attach to the language paradigm, it is clear that it has advantages and disadvantages when compared with the action paradigm implied by direct manipulation interfaces.
In the action paradigm, it is often much easier to **perform simple tasks without risk of certain classes of error**. For example, recognizing and pointing to an object reduces the difficulty of identification and the possibility of misidentification. On the other hand, more **complicated tasks are often rather tedious to perform in the action paradigm**, as they require repeated execution of the same procedure with only minor modification. In the language paradigm, there is the possibility of describing a generic procedure once (for example, a looping construct which will perform a routine manipulation on all files in a directory) and then leaving it to be executed without further user intervention.
The action and language paradigms need not be completely separate. An interesting combination of the two occurs in programming by example when a user can perform some routine tasks in the action paradigm and the system records this as a generic procedure. In a sense, the system is interpreting the user's actions as a language script which it can then follow.

4.2.9 Hypertext

In the context of scientific literature, where it is often very difficult to keep track of the origins and interrelations of the ever-growing body of research, a device which explicitly stored that information would be an invaluable asset.

Memex: A desk with the ability to produce and store a massive quantity of photographic copies of documented information.. Along with its huge storage capacity, it could keep track of links between parts of different documents.

Xanadu: Publishing and information retrieval system based on the idea of interconnected, non-linear text and other media forms. Ideas or footnotes within the text of a traditional paper(that is supposed to be read in linear fashion) that urge the reader to digress into a richer topic.

Hypermedia (or multimedia): Used for non-linear storage of all forms of electronic media. It reflects the use of non-linear and associative linking schemes.

4.2.10 Multi-modality

A multi-modal interactive system is a system that relies on the use of multiple human communication channels. In a sense, all interactive systems can be considered multi-modal as humans use visual and haptic channels while using them.
However, a genuine multi-modal system relies to a greater extent on simultaneous use of multiple communication channels for both input and output. We can refer to and point to someone in the real world using both voice and touch at the same time. Designers wanted to mimic this flexibility in both articulation and observation by extending inputs and outputs of interactive systems. Multimodal, multimedia and virtual reality systems form a large core of current research in interactive system design.

4.2.11 Computer supported cooperative work (CSCW)

Personal computing was all about providing individuals with enough computing power so that they were liberated from dumb terminals which operated on a time-sharing system. But after that, individuals wanted to reconnect themselves to their working environment and rest of the world.

CSCW vs Interactive Systems designed for single user: Designers can no longer neglect the society within which any single user operates. CSCW is built to allow interaction between humans and computers, so that it's possible to meet many people's needs.

Email(yet another metaphor): Instance of asynchronous CSCW. It's a communication via electronic message.

4.2.12 The world wide web (WWW)

Easy to use, predominantly graphical interface to info, hiding underlying complexities. The computers of the internet communicate using data transmission protocols and addressing systems. The web builds on this with http, html and urls.

Browsers allowed users to access multimedia info in some mouse clicks. Users realized that they can access the latest info instantly.

At present, the web is one of the major reasons for which new users are connecting to the internet. It is the reason why computers are considered as boxes that connect them with interesting people and places rather than being social contact denying soulless cases. This leads to the concept of "global village".

4.2.13 Agent-based interfaces

Software agents act on behalf of users (possibly, not necessarily, instructed in a linguistic fashion), performing repetitive tasks, watching and responding to events in the absence of the user and learning from the user's actions.

Challenges:
i) Developing a suitable language between human and agent when acting in user's absence.
ii) The user may not receive feedback of any mistake until effects become irreversible.

Examples:
i) Eager: Suggested repetitive actions to users working on simple HyperCad applications.
ii) Microsoft Excel: Intelligence in some useful arithmetic functions, such as summation. The intelligence in this is not embodied, it is diffuse, somewhere in the system.

Although **embodiment** is not essential to an intelligent agent. But it is one of the key features which enables users to determine where autonomy and intelligence may lie, and also which parts are stable.

4.2.14 Ubiquitous computing

In the 1980s, a group of researchers at Xerox PARC, led by Mark Weiser, initiated a research program with the goal of moving human–computer interaction away from the desktop and out into our everyday lives.

The intention of ubiquitous computing is to create a computing infrastructure that permeates our physical environment so that the computer becomes noticeable. For example, with the advancement, the utility of electronic motors led to ubiquity and hence invisibility from being a large, loud and noticeable machine.

Weiser thought that it was also important to think of computing technology in different sizes.
Three different scales of computing:
i) A yard-sized computer → Stanford interactive Mural
ii) A foot-sized computer → PC, Laptop, Tablet
iii) An inch-sized computer → PDA(Personal Digital Assistant), mobile phone, pocket dictionary

Waves of computing
i) First wave: one large mainframe computer served many people.
ii) Second wave: The PC revolution, computing devices roughly equaled the number of people using them.
iii) Third wave: Arrival of diverse computing devices

Ubiquitous computing is not simply about nifty gadgets, it is what can be done with those gadgets. Wireless networking, voice recognition, camera and vision systems, pen-based computing and positioning systems etc are converging to make the dream of ubiquitous computing possible.

Perks:
i) They provide the ability to move the computer user away from a desktop.
ii) Allow interaction in a number of modes (voice, gesture, handwriting) in addition to a keyboard
iii) Make information about the user (through vision, speech recognition or positioning information) available to a computational device that may be far removed from the actual user.

4.2.15 Sensor-based and context-aware interaction

The term ubiquitous computing encompasses a wide range from mobile devices to more pervasive environments. We can consider the extreme situation in which the user is totally unaware of interaction taking place. Information can be gathered from sensors in the environment. However, the vision of many is a world in which the whole environment is empowered to sense and even understand the context of activities within it.

In **context-aware computing** the interaction is more implicit. The sensor-enhanced environment is using heuristics and other semi-intelligent means to predict what would be useful for the user. The data used for this inference and the inference itself are both fuzzy, probabilistic and uncertain. So the actions resulting from intelligent predictions need to be made with caution.

Principles of appropriate intelligence:
i) Be right as often as possible, and useful when acting on these correct predictions.
ii) Do not cause inordinate problems in the event of an action resulting from a wrong prediction.

The failure of 'intelligent' systems in the past resulted from following the first principle, but not the second.

Arguably this is a **more radical paradigm shift than any other** since the introduction of interactive computing itself.
Ubiquitous vs Context-aware: Ubiquitous computing challenges the idea of where computers are and how apparent they are to us, context-aware computing challenges what it means to interact with a computer.

Summary

5.1 INTRODUCTION

Interaction design is not just about the artifact 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**.

It may be better not to think of designing a system, or an artifact,
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!

5.2 WHAT IS DESIGN

achieving goals within constraints
This does not capture everything about design but helps to focus us on certain
things:

a) **Goals:** What is the purpose of the design we are intending to produce? Who is it for? Why do they want it?

b) **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?

c) **Trade-off:** Choosing which goals or constraints can be relaxed so that others can be met. The temptation is to focus on one or another 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.

5.2.1 The Golden Rule of Design

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

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

1) **understand computers:** limitations, capacities, tools, platforms
2) **understand people:** psychological, social aspects, human error.

5.2.2 To err is human

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 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.

5.2.3 The central message – the user

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

5.3 THE PROCESS OF DESIGN

In the best companies, however, usability is designed from the start.

Here we'll take a simplified view of **four main phases plus an iteration loop**, focusing on the design of interaction (Figure 5.1)
*** see figure 5.1 from book**

1) **Requirements – what is wanted:** *how do people currently watch movies? What sort of personal appliances do they currently use?*

There are several techniques used for this in HCI: **interviewing people, videotaping them, looking at the documents and objects that they work with, observing them directly.** In particular, **ethnography**, a form of observation deriving from anthropology, has become very influential.

2) **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. We will look at

scenarios, rich stories of interaction, which can be used in conjunction with a method like a **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.**

3) **Design:** 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. It is at this stage also where input from theoretical work is most helpful, including **cognitive models, organizational issues, and understanding communication**

4) **Iteration and prototyping:** Humans are complex and we cannot expect to get designs right the first time. We, therefore, need to evaluate a design to see **how well it is working and where there can be improvements**. 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**.

5) **Implementation and deployment:** 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.

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 tradeoffs, it may be possible to find radically different solutions that have a major effect but are cheap to implement.

5.4 USER FOCUS

The start of any interaction design exercise must be the intended user or users. This is often stated as:
know your users

Over time many people are affected directly or indirectly by a system and these people are called stakeholders. So, how to know about the users:

1) **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? 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.

2) **Probably not like you!** When designing a system it is easy to design it as if you were the main user: you assume your 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.

3) **Talk to 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. By involving users throughout the design process it is possible to get a deep knowledge of their work context and needs.

4) **Watch them:** Although what people tell you is of the utmost importance, it is not the whole story. 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 art 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.

5) **Use your imagination:** 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. *** see figure 5.3 from the book**

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.** 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.**

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 **the plain text**, but scenarios can be augmented by **sketches, simulated screenshots, etc.** 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?'.**

In addition, scenarios can be used to:

1) **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.
2) **Validate other models:** A detailed scenario can be 'played' against various more formal representations such as task models or dialog and navigation models
3) **Express dynamics:** Individual screenshots and pictures give you a sense of what a the system would look like, but not how it behaves.

This linearity property of the scenario has both positive and negative points:

1) **Positive – Time is linear:** Our lives are linear as we live in a time and so we find it easier to understand simple linear narratives. We are natural storytellers and story listeners.
2) **Negative – 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.

5.6 NAVIGATION DESIGN

We interact at several levels:

Table 5.1 Levels of interaction

PC application	Website	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!

The place to start when considering the structure of an application is to think about actual use:
1) who is going to use the application?
2) how do they think about it?
3) what will they do with it?

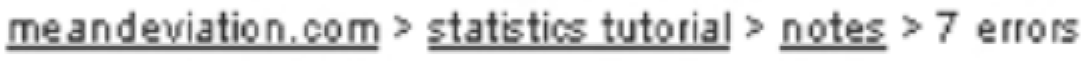
Here we will consider two main kinds of issues:

1) **local structure:** looking from one screen or page out, individual page
2) **global structure:** structure of the whole site, movement between the screens.

5.6.1 Local Structure

Here are **four things** to look for when looking at a single web page, screen, or state of a device.

1) **knowing where you are:**



meandeviation.com > statistics tutorial > notes > 7 errors

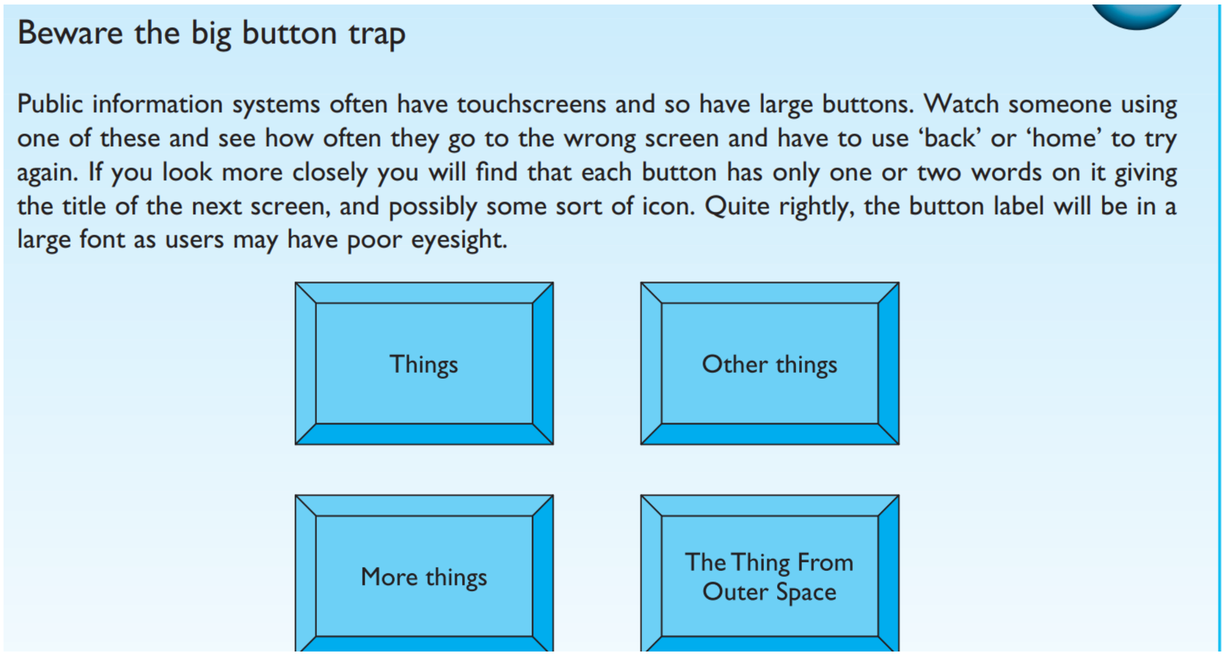
Showing the **current path with the navigation paths** at the top of the web page.

2) **knowing what you can do:** Using buttons with proper choice of button name on it so that the user can understand easily what will happen by pressing the button. The clickable links by underlining into the words. Not using **confusing notations in the passive components** (like putting an underline under a word to empathize it but users misunderstand it as a clickable link)
3) **knowing where you are going – or what will happen:** Making the clickable **links or buttons self-explanatory** so that users understand what will be the

consequences after pressing or clicking it without trial and error approach

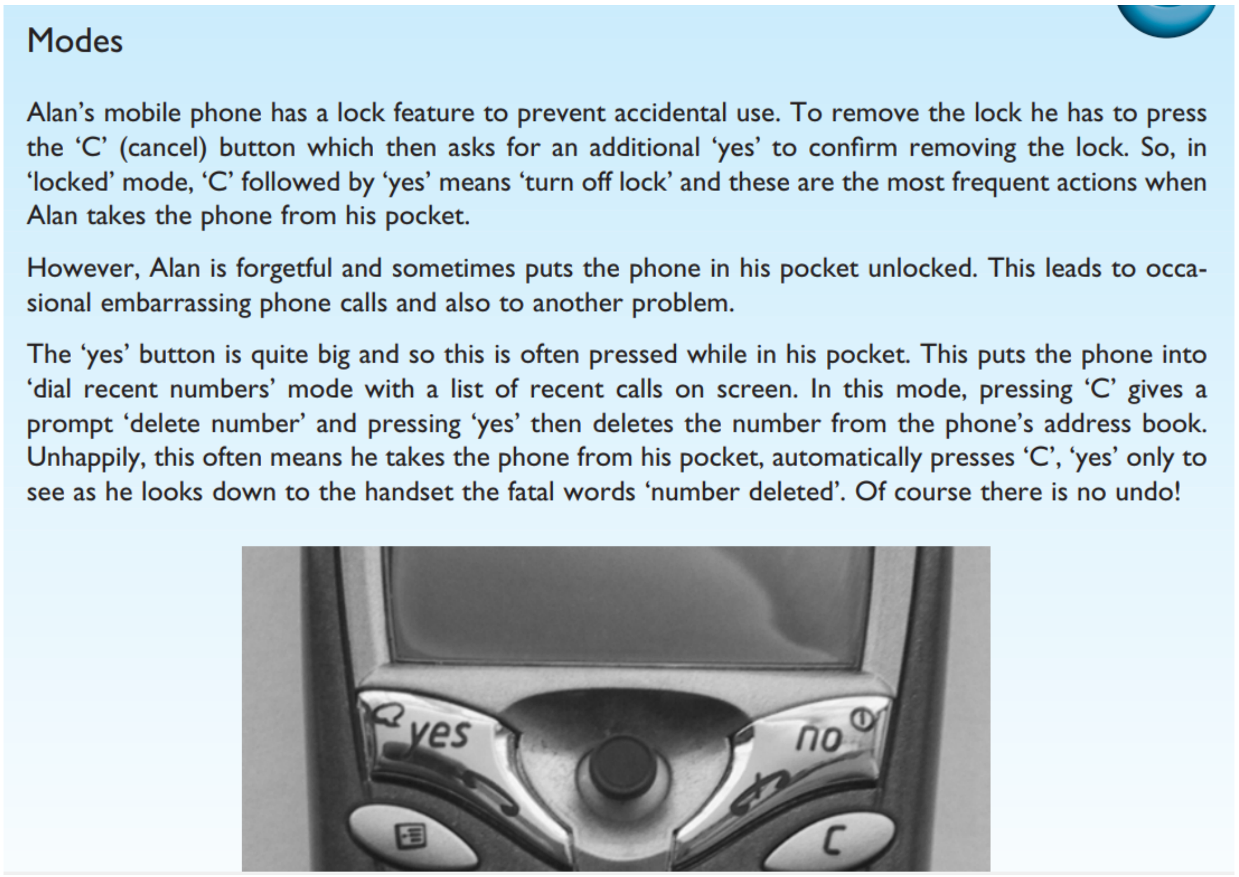
4) **knowing where you've been – or what you've done:** Having a **back button or backing option** to the previous page/ state so that users can understand the navigation and its hierarchy easily

Design focus:



Beware the big button trap

Public information systems often have touchscreens and so have large buttons. Watch someone using one of these and see 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.

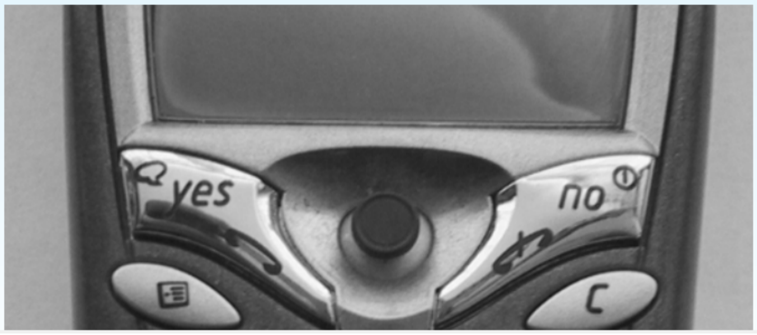


Modes

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 address 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!



5.6.1 Global structure – hierarchical organization

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.**

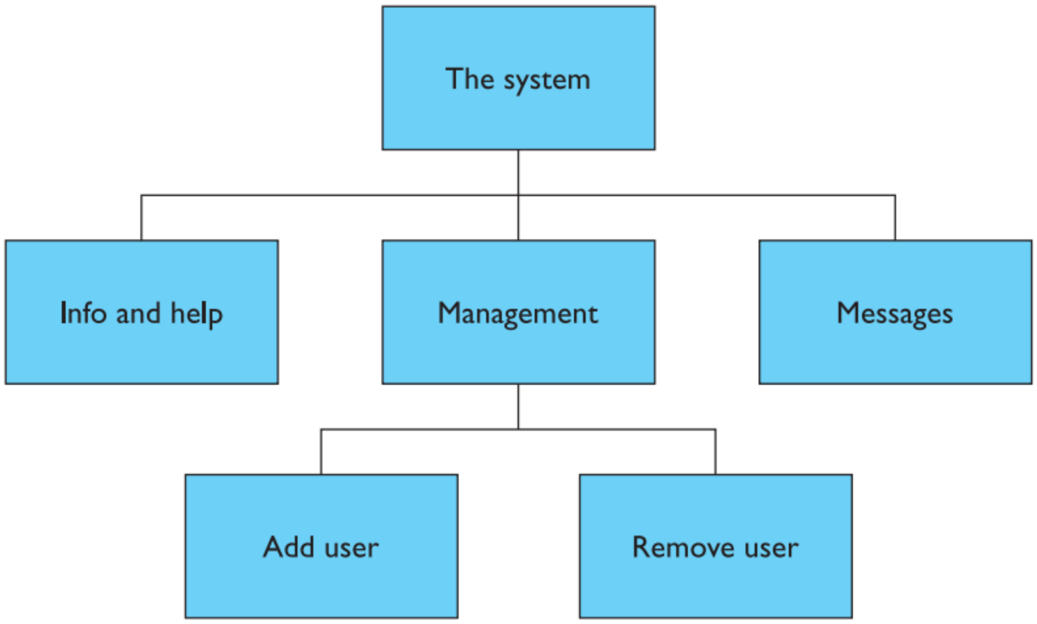


Figure 5.6 Application functional hierarchy

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 the menu on one screen or web page. Miller's magic number of 7 ± 2 for working memory capacity 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. But in the real world, it depends on the purpose of the system and the understanding capacities of the user groups as well. The menus can be from ¾ to 60! There should be no restrictions about it.

5.6.3 Global structure – dialogue

In HCI the word **'dialog'** is used to refer to this pattern of interactions between the user and a system.

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

1) **show what leads to what**
2) **show what happens when**
3) **include branches and loops**
4) **be more task-oriented than a hierarchy**

Figure 5.7 shows a network diagram illustrating 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.

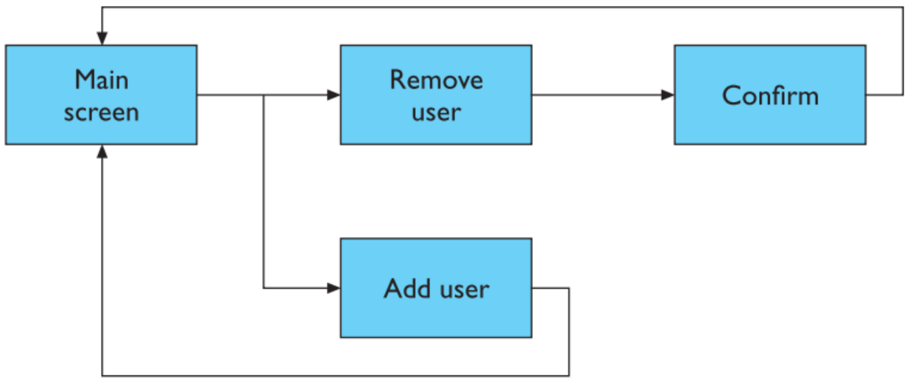


Figure 5.7 Network of screens/states

5.6.4 Wider still

Some other things to keep in mind while designing:

1) **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 and pause icons.

2) **Functional issues:** On a PC application we need to be able to interact with files, read standard formats and be able to handle cut and paste.

3) **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 the application was launched for the attachment.

5.7 SCREEN DESIGN AND LAYOUT

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

1) **Ask:** What is the user doing?

2) **Think:** What information is required? What comparisons may the user need to make? In what order are things likely to be needed?

3) **Design:** Form follows function: let the required interactions drive the layout.

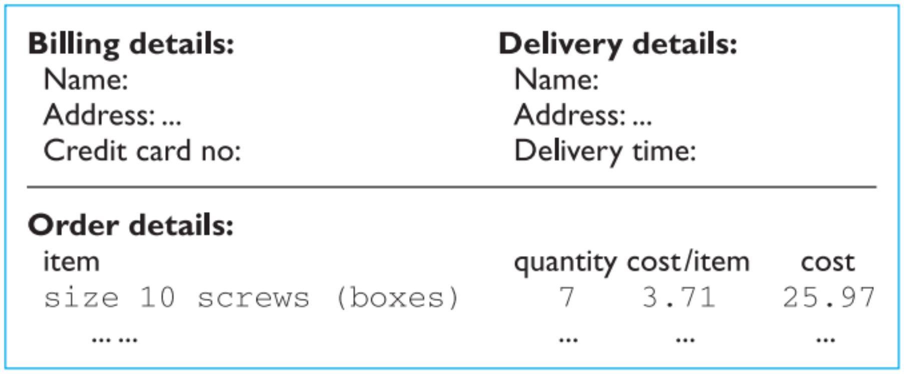
5.7.1 Tools for layout

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

1) Grouping and structure: If things logically belong together, then we should normally physically group them. 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 spatially

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

3) **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 colors can be used to emphasize groupings.



Billing details:	**Delivery details:**
Name:	Name:
Address: ...	Address: ...
Credit card no:	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

4) **Alignments:** The 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 should normally be aligned to the right (for integers) or at the decimal**

point. This is because the shape of the column then indicates the magnitude. Consider list **(i)** in Figure 5.10. It is 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).**

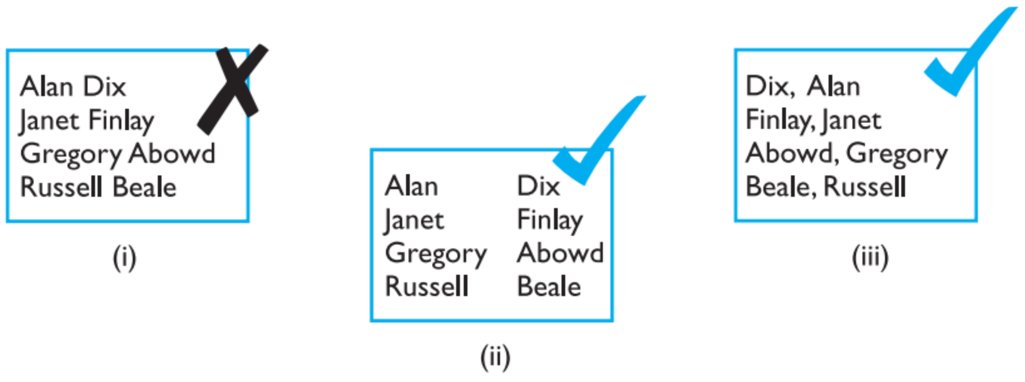
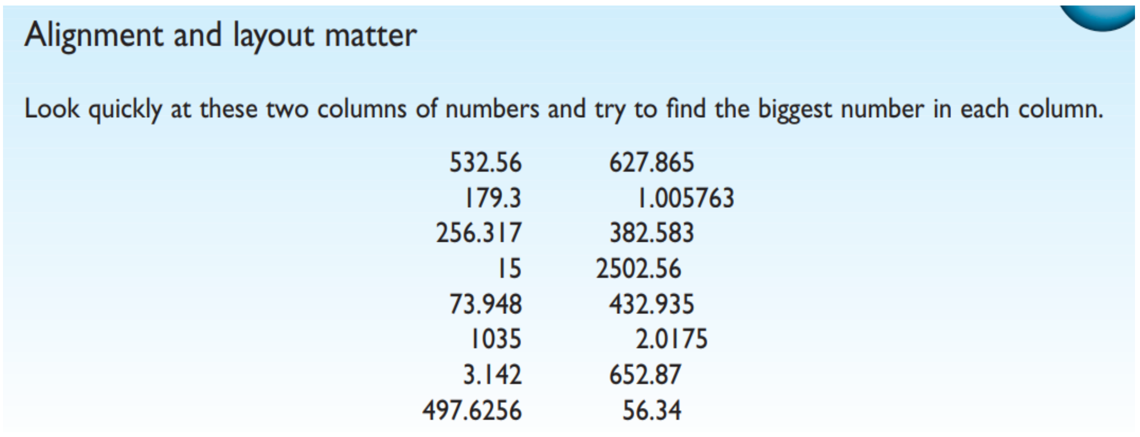


Figure 5.10 Looking up surnames

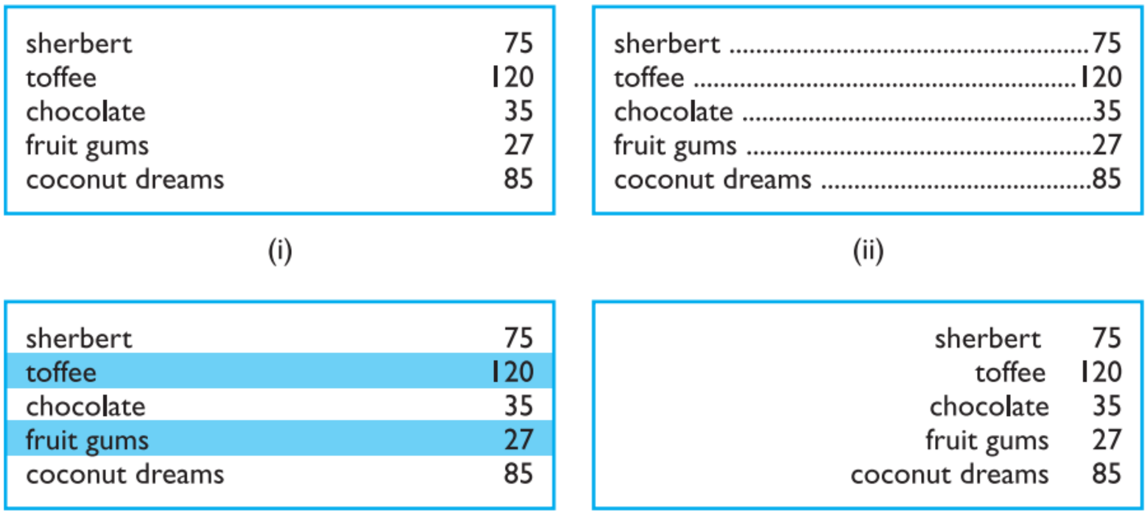


Alignment and layout matter

Look quickly at these two columns of numbers and try to find the biggest number in each column.

```
   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
```

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 grays or colors** behind rows or columns. This 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 frequently to 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 versus the ability to see relationships across rows



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

(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

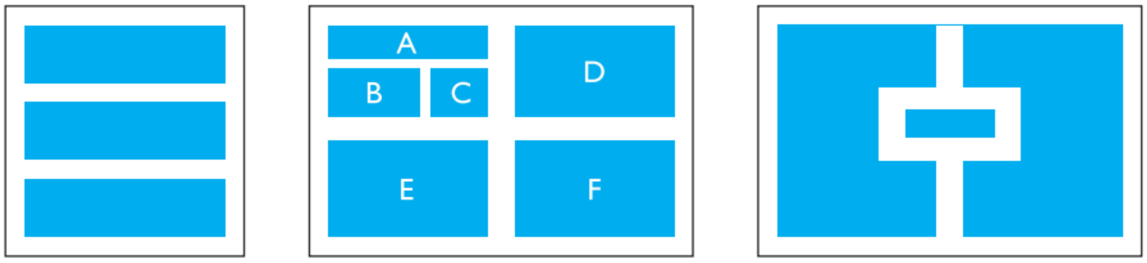
(iii)

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

(iv)

Figure 5.11 Managing multiple columns

5) White scape: In typography, the space between the letters is called **the counter.** Space can be used in several ways. Some of these are shown in **Figure 5.12**. The colored 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 their own, followed by B and C together. **In Figure 5.12 (iii)**, we can see the space used to highlight. This is a technique used frequently in magazines to highlight a quote or graphic.

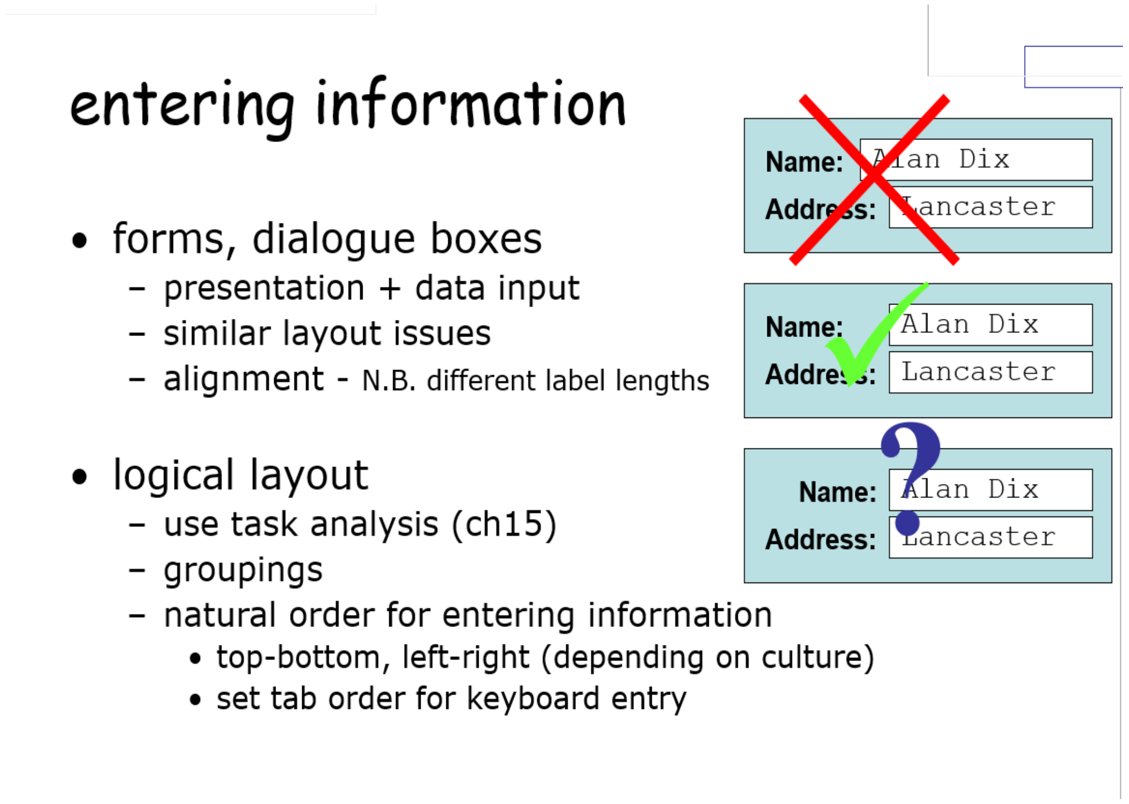


(i) Space to separate (ii) Space to structure (iii) Space to highlight

Figure 5.12 Using white space in layout

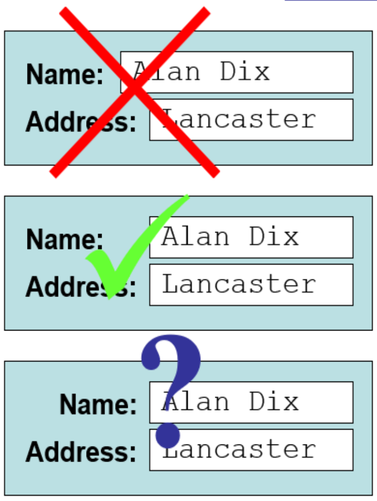
5.7.2 User action and control

1) Entering information:



entering information

- forms, dialogue boxes
  - presentation + data input
  - similar layout issues
  - alignment - N.B. different label lengths
- logical layout
  - use task analysis (ch15)
  - groupings
  - natural order for entering information
    - top-bottom, left-right (depending on culture)
    - set tab order for keyboard entry



2) Knowing what to do:

knowing what to do

- what is active what is passive
  - where do you click
  - where do you type
- consistent style helps
  - e.g. web underlined links
- labels and icons
  - standards for common actions
  - language – bold = current state or action

3) Affordances:

affordances

- psychological term
- for physical objects
  - shape and size suggest actions
    - pick up, twist, throw
  - also cultural – buttons 'afford' pushing
- for screen objects
  - button–like object 'affords' mouse click
  - physical-like objects suggest use
- culture of computer use
  - icons 'afford' clicking
  - or even double clicking … not like real buttons!

5.7.3 Appropriate appearance

1) 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**. 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.

2) **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

3) Making a mess of color and 3D shapes: One of the worst features in many interfaces is their appalling use of color. If the **graphics card of the user's computer is not good, a color or 3D modeling can be shown weirdly on that users' PC.** It will lose the appeal of the site and the usability will be hampered to a great extent.

4) Localization/ internationalization: The process of making software suitable for different languages and cultures is called localization or internationalization.
However, changing the language is only the simplest part of internationalization.
Much of the explicit guidance on alignment and layout is dependent on **left-to-right, top-to-bottom languages such as English and most European languages.**
Furthermore, many icons and images are only meaningful within a restricted cultural context. One cannot simply assume that all the 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

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.**

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 hilltop. 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 on doing this until it can't get any better.

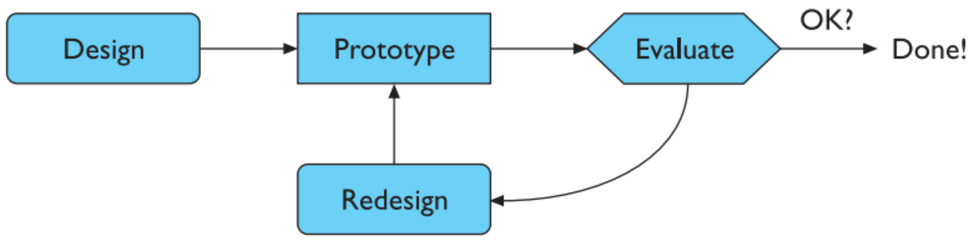


Figure 5.14 Role of prototyping

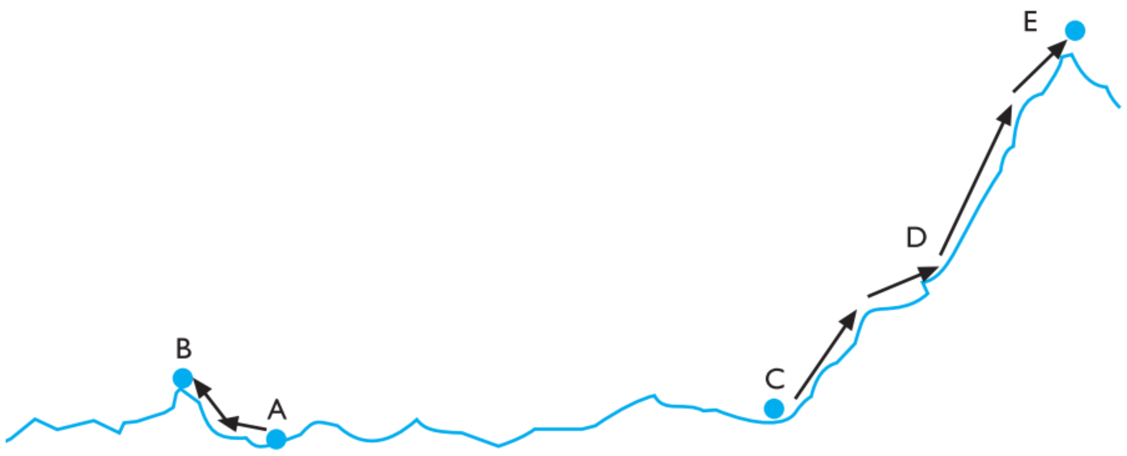


Figure 5.15 Moving little by little . . . but to where?

Figure 5.15 shows this schematically: if you start at A you get trapped at the local maximum at B, but if you start at C you move up through D to the global maximum at E. 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:

1) To understand what is wrong and how to improve: you cannot iterate the design unless you know what must be done to improve it.

2) A good start point: A good start point will help to reach *the goal or the global maxima* in a shorter time.

A really good designer might guess a good initial design based on **experience and judgment.** 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.