UNIT 2 - DESIGN & SOFTWARE PROCESS

Interactive Design basics – process – scenarios – navigation – screen design – Iteration and prototyping. HCI in software process – software life cycle – usability engineering – Prototyping in practice – design rationale. Design rules – principles, standards, guidelines, rules. Evaluation Techniques – Universal Design.

2.1 INTERACTIVE DESIGN BASICS

A simple definition of design is: achieving goals within constraints

- goals - purpose
  - Who is it for, why do they want it
- constraints
  - Materials, platforms used
- trade-offs

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 while 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.

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:
  - o understand computers
    – limitations, capacities, tools, platforms
  - o understand people
    – psychological, social aspects, human error.

To err is human

- 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.
- 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

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

2.2 THE PROCESS OF DESIGN

- 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. Simplified view of four main phases plus an iteration loop, focused on the design of interaction.

  a. **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. 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
  b. **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.
  c. **Design**: 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.
  d. **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
  e. **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

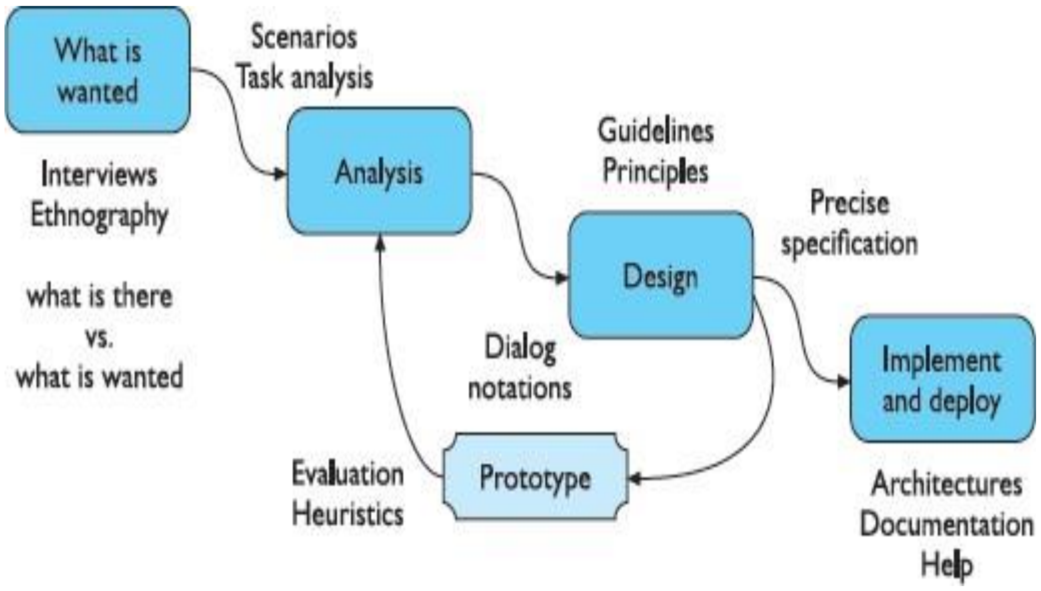


Fig 2.1: Interface Design Process

2.2.1 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 user

Know your user

- The start of any interaction design exercise must be the intended user or users. This is often stated as: know your users. So, how do you get to know your users?
    - **Who are they?:** The first thing to find out is who your users are. Are they young or old, experienced computer users or novices?
    - **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
    - **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
    - **Watch them:** 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.

Persona

- One method that has been quite successful in helping design teamsproduce user focused designs is the persona.
- the aspect of someone's character that is presented to or perceived byothers
- description of an 'example' user
    - not necessarily a real person
- use as substitute user

example persona

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 interrupted 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).

Cultural probes

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 includes a glass with a paper sleeve

- direct observation sometimes hard
    - in the home
    - psychiatric patients, …
- probe packs are introduced with items to prompt responses
    - e.g. glass to listen at wall, camera, postcard
- given to people to open in their own environment
  they record what is meaningful to them
- used to …
    - inform interviews, prompt ideas, enculture designers

2.3 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. This gives answers for the following questions
    - what will users want to do?
    - step-by-step walkthrough
        - what can they see (sketches, screen shots)
        - what do they do (keyboard, mouse etc.)
        - what are they thinking?
    - use and reuse throughout design

scenario – movie player

Brian would like to see the new film “Moments of Significance” and wants to invite Alison, 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 compatible 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 whirling icon. … … …

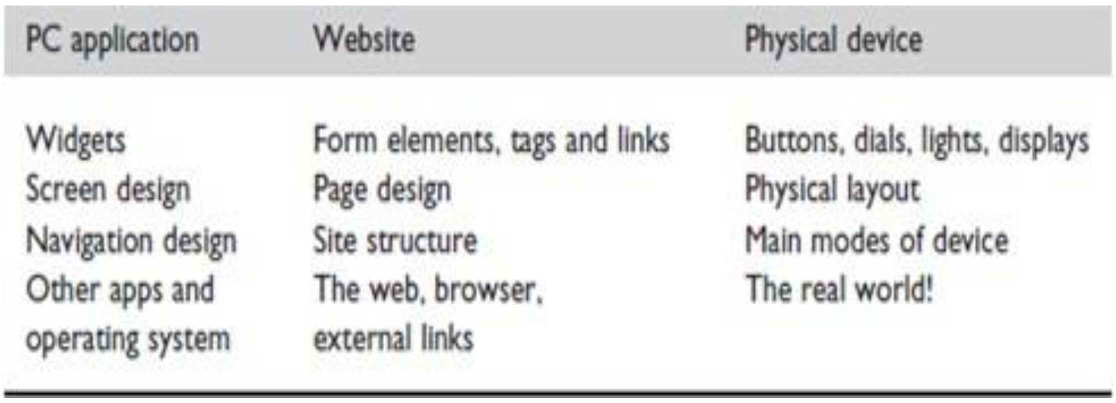
Scenarios can be used to:

i) 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.
ii) Validate other models A detailed scenario can be ‘played’ against various more formal representations such as task models or dialog
and navigation models.
iii) Express dynamics Individual screen shots and pictures give you a sense of what a system would look like, but not how it behaves. This linearity has both positive and negative points:
iv) 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.
v) 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.

2.4 NAVIGATION

- 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. But now we are focusing on the computer system itself. You interact at several levels:

a. 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.
b. Screens or windows You need to find things on the screen, understand the logical grouping of buttons.
c. 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.



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!

d. Environment The word processor has to read documents from disk, perhaps some are on remote networks. You swap between applications, perhaps cut and paste.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.

2.4.1 Local Structure

- Much of interaction involves goal-seeking behavior. 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.
- 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. 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.

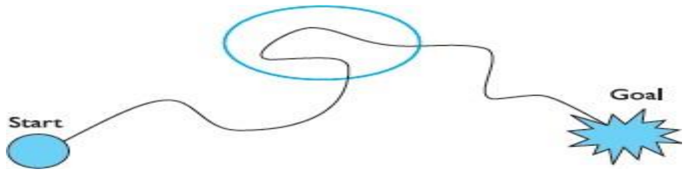
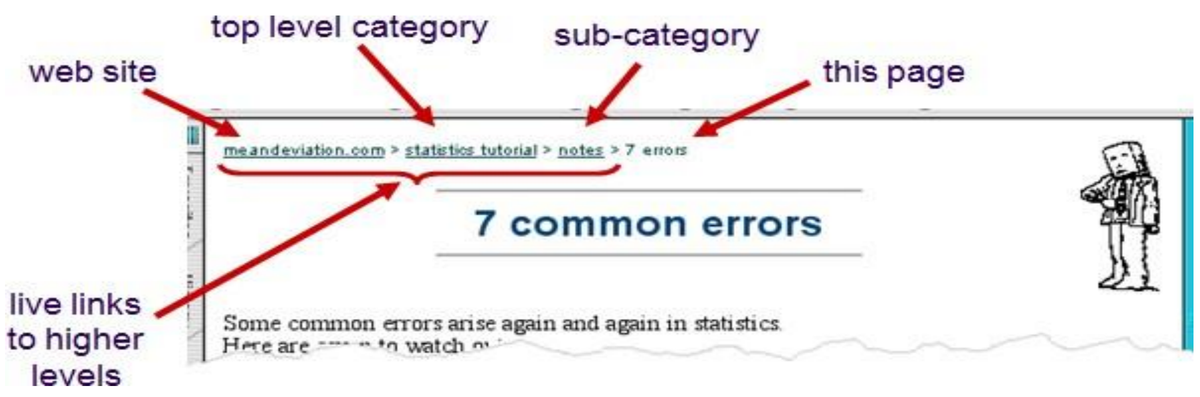


Fig.2.2 Local Structure

Four golden rules

- 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

Where you are

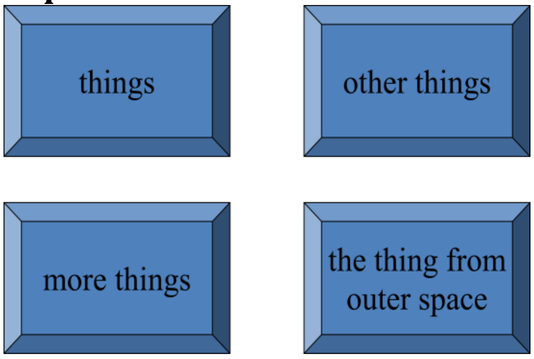

top level category
sub-category
web site
this page
live links to higher levels
meandeviation.com > statistics tutorial > notes > 7 errors

7 common errors

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

navigation system that shows a user's location in a site or web app.

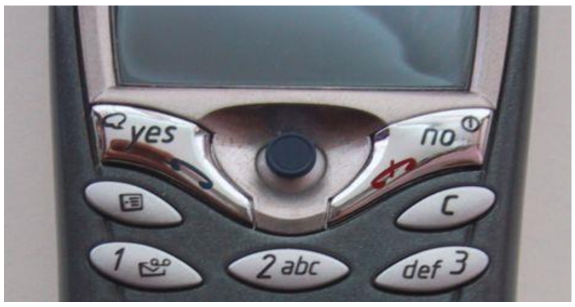
beware the big button trap



Public information systems often have touch screens 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

Modes

- lock to prevent accidental use ...
  - remove lock - 'c' + 'yes' to confirm
  - frequent practiced action
- if lock forgotten
  - in pocket 'yes' gets pressed
  - goes to phone book
  - in phone book ...
    'c' – delete entry
    'yes' – confirm
    ... oops !



2.4.2 Global Structure – Hierarchical Organization

- 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 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.
- Any sort of information structuring is difficult, but there is evidence that people find hierarchies simpler than most. One 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.

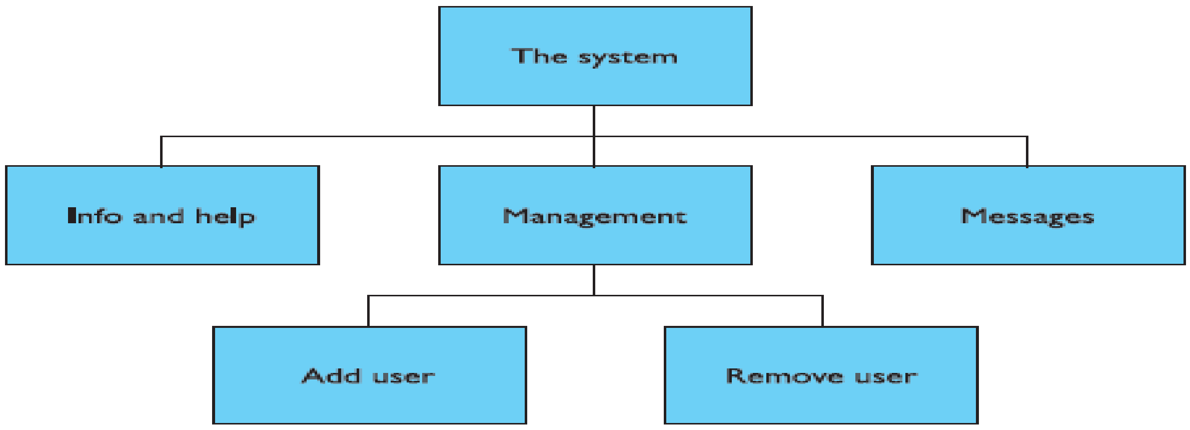


Fig 2.3: Application Functional Hierarchy

2.4.3 Global structure – dialog

- A pure information system or static website it may be sufficient to have a fully hierarchical structure, perhaps with next/previous links between items in the same group.
- A 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.
- 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 'dialog' 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 dialog 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. A simple way is to use a network diagram showing the principal states or screens. The 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.

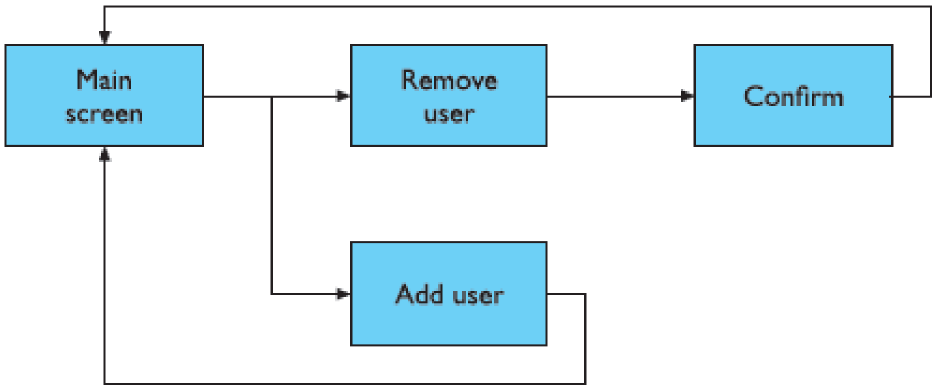


Fig 2.4: Network of screens/states

- It shows a network diagram illustrating the main screens for adding or deleting a user from the messaging system in Figure . 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.

2.4.4 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:

a. 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.
b. 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.
c. 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.

2.5 SCREEN DESIGN AND LAYOUT

- 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 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.

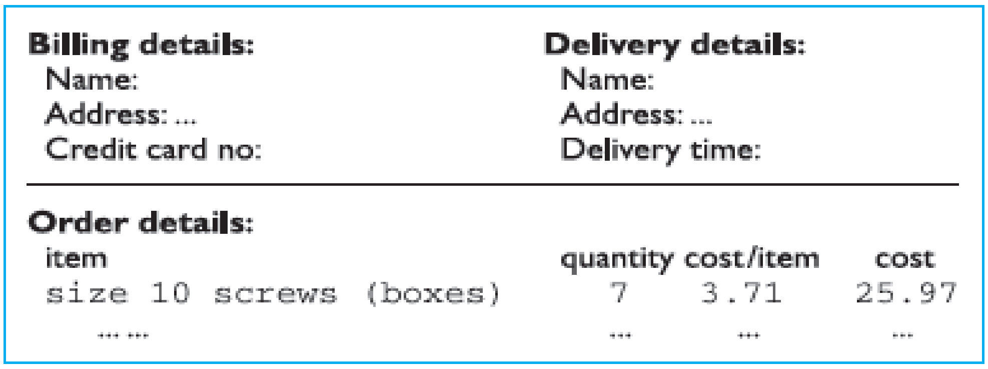
2.5.1 Tools for layout

- 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.

- Available tools
  - **Grouping of items**
  - **Order of items**
  - **Decoration - fonts, boxes etc.**
  - **Alignment of items**
  - **White space between items**

a. 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 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 spatially.



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
... ...	...	...	...

Fig 2.5: Grouping Related Items in an order screen

This reflects the following logical structure:

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

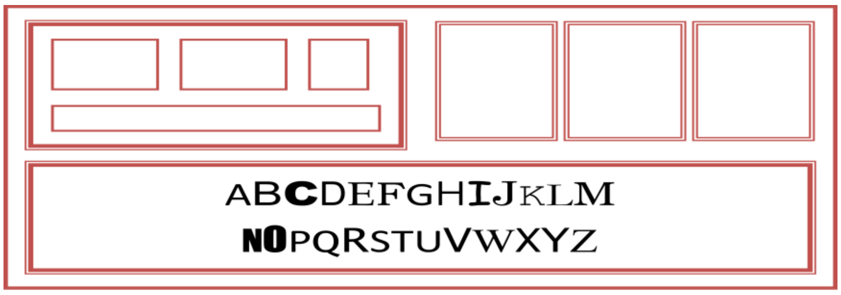
b. Order of groups and items

- If we look at above Figure again we can see that 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 dial

c. Decoration

- 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. See how the buttons differ in using the foreground and back-ground colors (green and gold) so that groups are associated with one another.

- use boxes to group logical items
- use fonts for emphasis, headings



d. Alignment

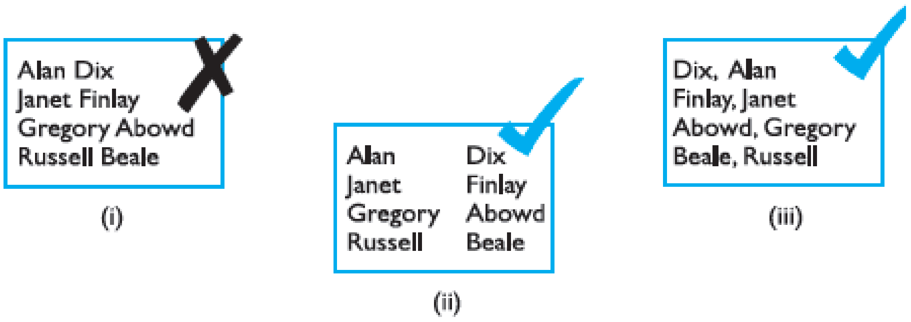
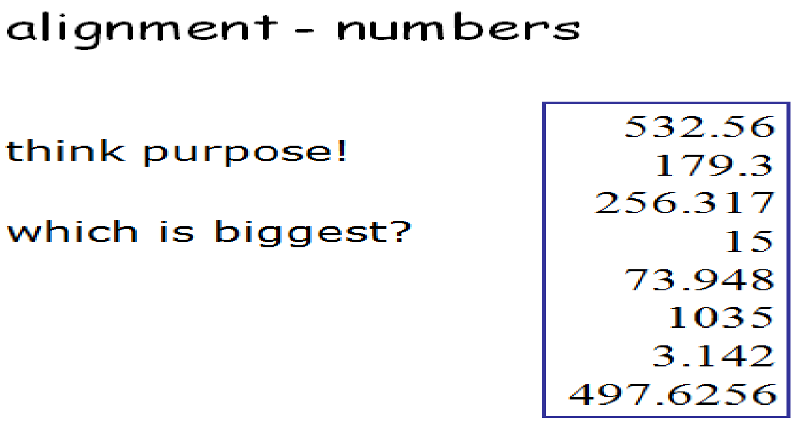
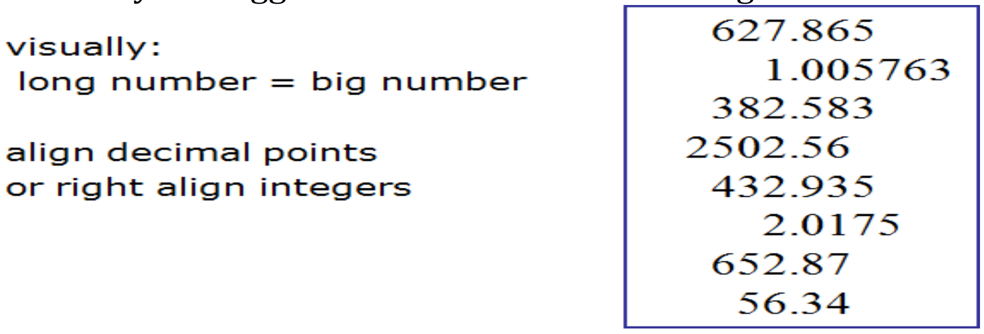


Fig 2.6: Looking up surnames

Alignment - numbers

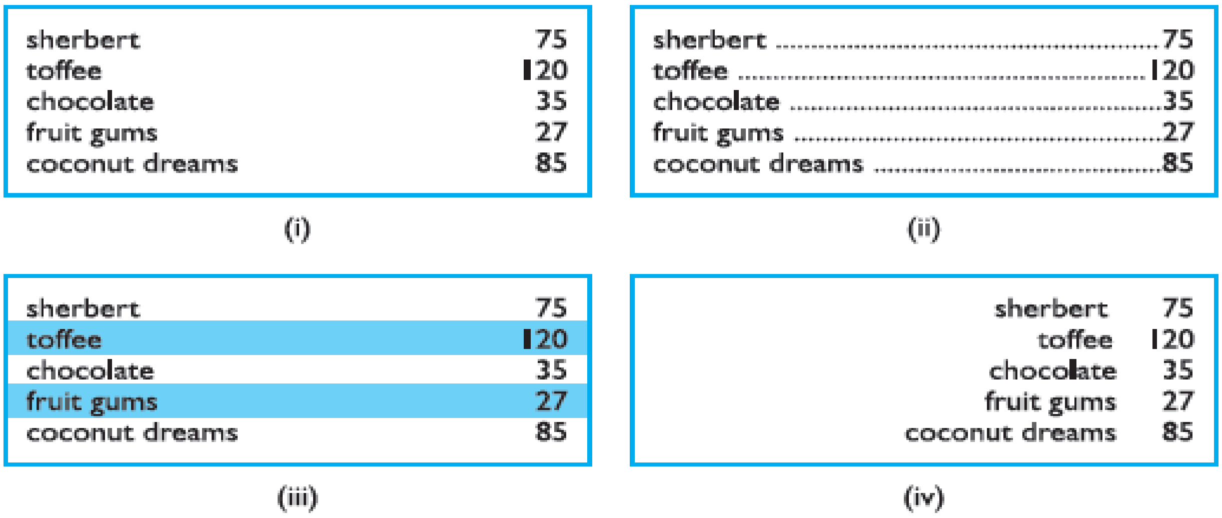


Difficult to identify the biggest number because of un aligned numbers.



- 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. 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 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



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)

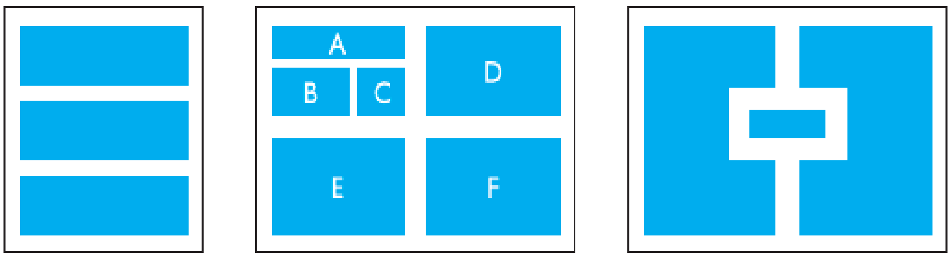
sherbert	75
toffee	120
chocolate	35
fruit gums	27
coconut dreams	85

(iv)

Fig 2.7: Managing Multiple Columns

e. White space

- 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 in several ways. Some of these are shown in Figure The colored areas represent continuous areas of text or graphics. (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.



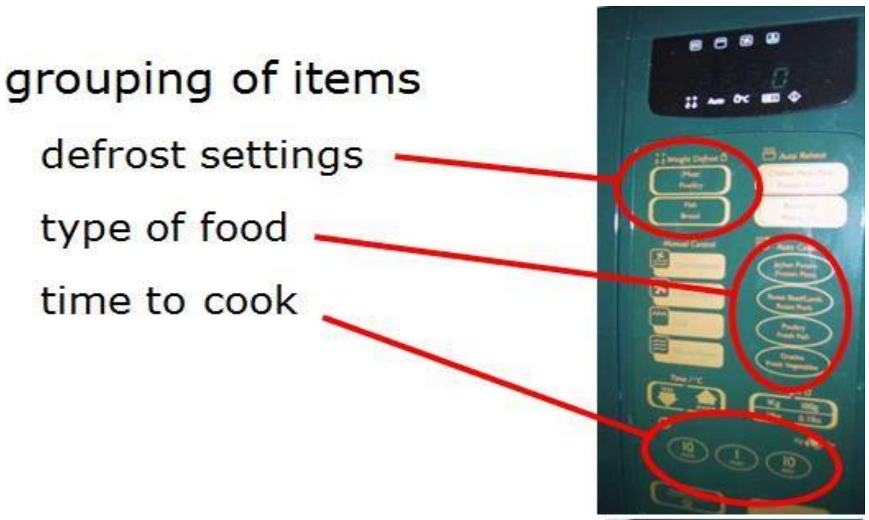
(i) Space to separate (ii) Space to structure (iii) Space to highlight

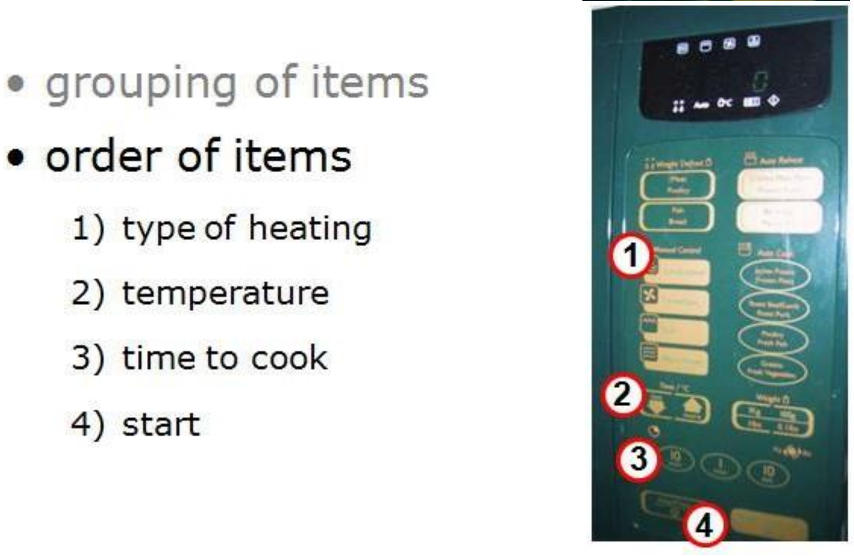
Fig 2.8: Using white space in layout

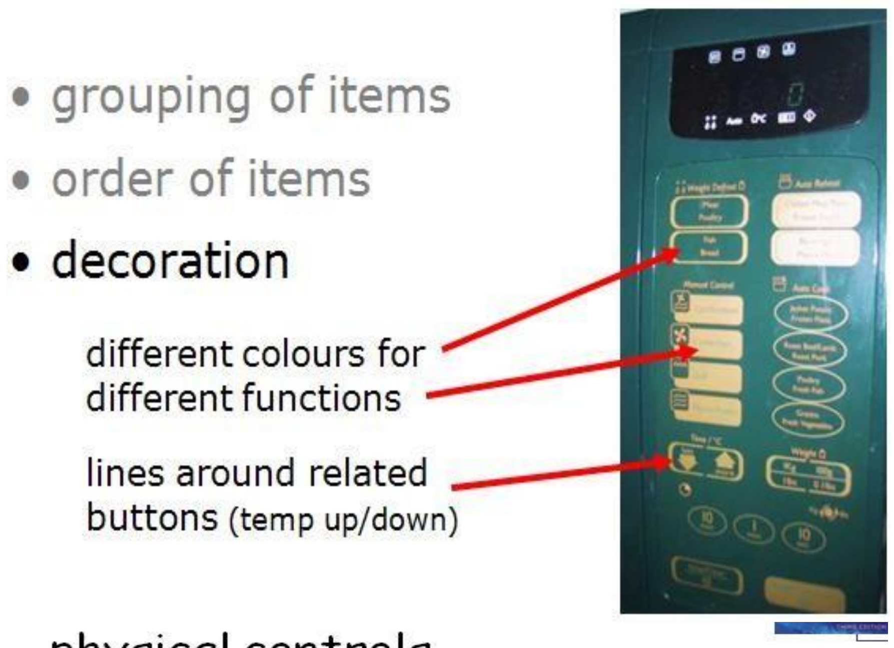
(ii) There are clearly four main areas: ABC, D, E and F. Within one of these are three further areas, A, B and C, which themselves are grouped as A on its own, followed by B and C together.

(iii) We can see space used to highlight. This is a technique used frequently in magazines to highlight a quote or graphic.

Example : physical controls in Microwave control panel



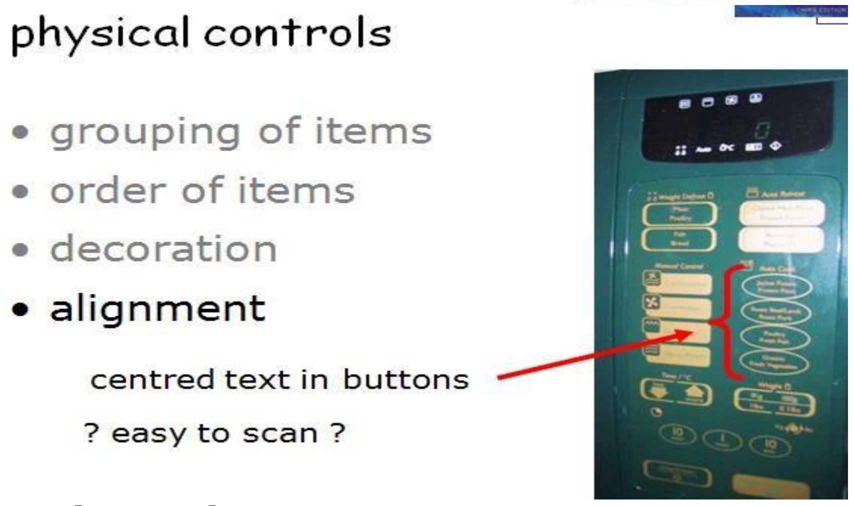




- grouping of items
- order of items
- **decoration**

different colours for different functions

lines around related buttons (temp up/down)



physical controls

- grouping of items
- order of items
- decoration
- **alignment**

centred text in buttons

? easy to scan ?

2.5.2 User Action And Control

a. 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 userto 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.

b. 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 to look the same and menus to look the same, then users will be able to recognize them when they see them.

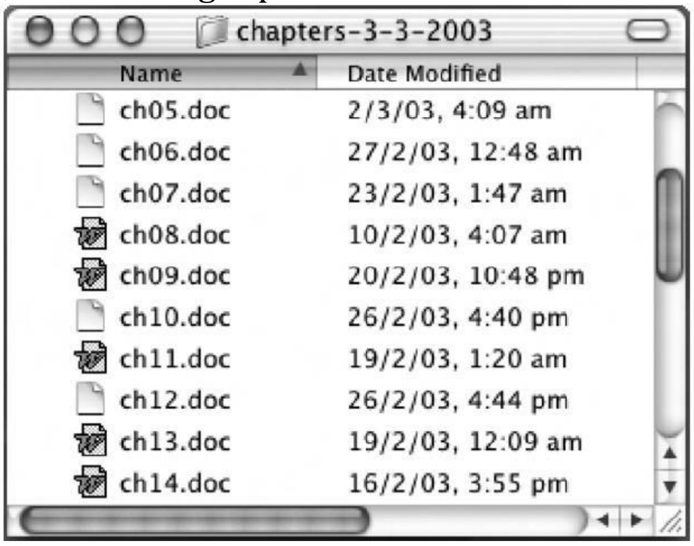
c. Affordances

- These are especially difficult problems in multimedia applications where one may deliberately adopt a non-standard and avant-garde style. 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.

2.5.3 Appropriate appearance

a. 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.
- 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.
- 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.



chapters-3-3-2003

Name	Date Modified
ch05.doc	2/3/03, 4:09 am
ch06.doc	27/2/03, 12:48 am
ch07.doc	23/2/03, 1:47 am
ch08.doc	10/2/03, 4:07 am
ch09.doc	20/2/03, 10:48 pm
ch10.doc	26/2/03, 4:40 pm
ch11.doc	19/2/03, 1:20 am
ch12.doc	26/2/03, 4:44 pm
ch13.doc	19/2/03, 12:09 am
ch14.doc	16/2/03, 3:55 pm

Alphabetic file listing.

b. Aesthetics and utility

- 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.

c. Making a mess of it: color and 3D

- One of the worst features in many interfaces is their appalling use of color. This is partly because many monitors only support a limited range of primary colors 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 color can be distracting that a significant proportion of the population is color blind, may mean that parts of the text are literally invisible to some users. In general, color 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.

d. Localization / internationalization

- If you are working in a different country, you might 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.

2.6 ITERATION AND PROTOTYPING

- This often starts early on with paper designs and storyboards being demonstrated to colleagues and potential users. 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. 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.
- The Figure shows this process. The end point is when there are no more problems that can economically be fixed. 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.

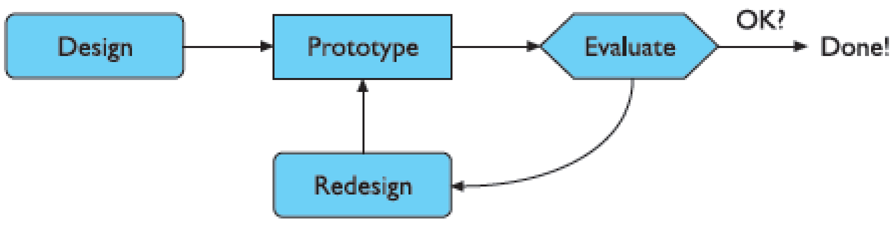
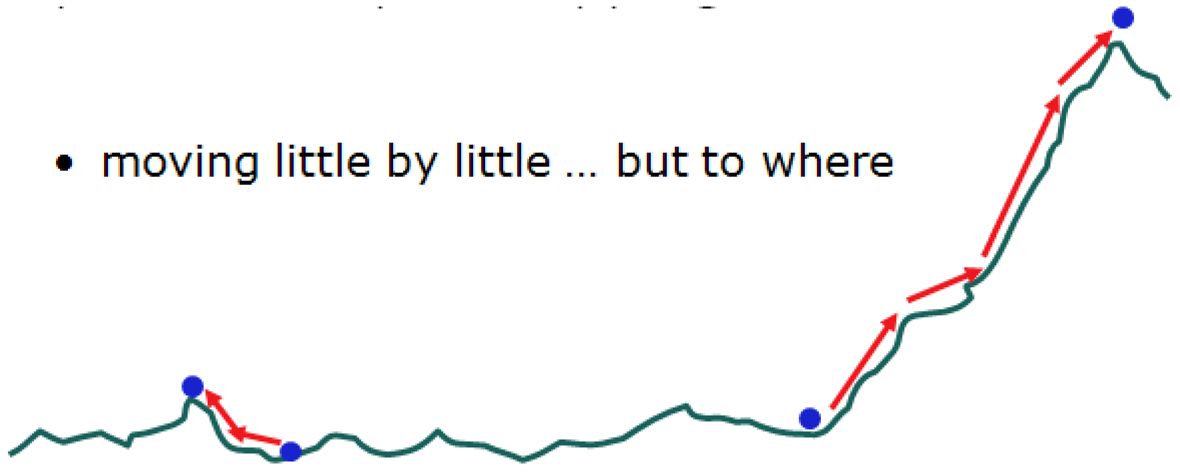


Fig 2.9: Role of Prototyping

- 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.
  2. 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 wanted to climb as high as you could, you would probably book 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.
- pitfalls of prototyping
  - 1. need a good start point
  - 2. need to understand what is wrong



2.7 HCI IN SOFTWARE PROCESS

- The design goal is to provide reliable techniques for the repeated design of successful and usable interactive systems. It is therefore necessary that we go beyond the exercise of identifying paradigms and examine the process of interactive system design.
- Within computer science there is already a large sub discipline that addresses the management and technical issues of the development of software systems – called software engineering. One of the cornerstones of software engineering is the software life cycle, which describes the activities that take place from the initial concept formation for a software system up until its eventual phasing out and replacement.
- The important point that we would like to draw out is that issues from HCI affecting the usability of interactive systems are relevant within all the activities of the software life cycle. Therefore, software engineering for interactive system design is not simply a matter of adding one more activity that slots in nicely with the existing activities in the life cycle. Rather, it involves techniques that span the entire life cycle.

2.8 SOFTWARE LIFE CYCLE

- A fundamental feature of software engineering, therefore, is that it provides the structure for applying techniques to develop software systems. The software life cycle is an attempt to identify the activities that occur in software development. These activities must then be ordered in time in any development project and appropriate techniques must be adopted to carry them through.
- In the development of a software product, we consider two main parties: the customer who requires the use of the product and the designer who must provide the product. Typically, the customer and the designer are groups of people and some people can be both customer and designer. It is often important to distinguish between the customer who is the client of the designing company and the customer who is the eventual user of the system.
- These two roles of customer can be played by different people. The group of people who negotiate the features of the intended system with the designer may never be actual users of the system. This is often particularly true of web applications**.**

2.8.1 Activities in the life cycle

- The graphical representation is reminiscent of a waterfall, in which each activity naturally leads into the next. The analogy of the waterfall is not completely faithful to the real relationship between these activities, but it provides a good starting point.

Requirements specification

- It involves eliciting information from the customer about the work environment, or domain, in which the final product will function. Aspects of the work domain include not only the particular functions that the software product must perform but also details about the environment in which it must operate, such as the people whom it will potentially affect and the new product’s relationship to any other products which it is updating or replacing. It begins at the start of product development.
- Though the requirements are from the customer’s perspective, if they are to be met by the software product they must be formulated in a language suitable for implementation.
- Requirements are usually initially expressed in the native language of the customer. The executable languages for software are less natural and are more closely related to a mathematical language in which each term in the language has a precise interpretation, or semantics.

Architectural design

- The next activities concentrate on how the system provides the services expected from it. The first activity is a high-level decomposition of the system into components that can either be brought in from existing software products or be developed from scratch independently.
- An architectural design performs this decomposition. It is not only concerned with the functional decomposition of the system, determining which components provide which services. It must also describe the interdependencies between separate components and the sharing of resources that will arise between components.

- There are many structured techniques that are used to assist a designer in deriving an architectural description from information in the requirements specification (such as CORE, MASCOT and HOOD).

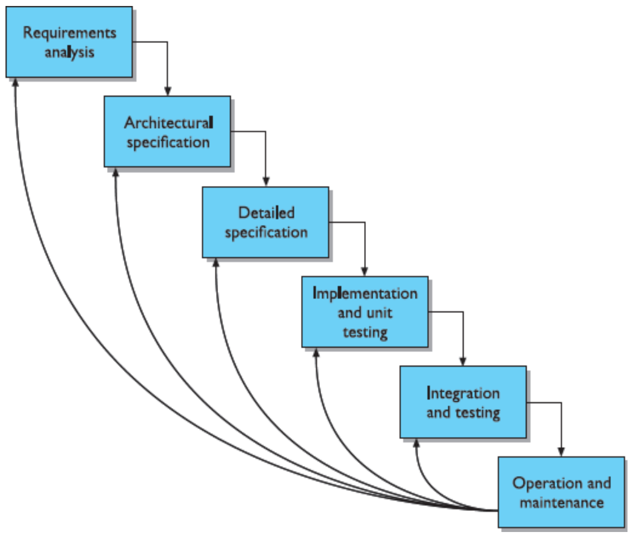

Requirements analysis
Architectural specification
Detailed specification
Implementation and unit testing
Integration and testing
Operation and maintenance

Fig 2.10: Feedback from maintenance activity to other design activity

Detailed design

- The detailed design is a refinement of the component description provided by the architectural design. The behavior implied by the higher-level description must be preserved in the more detailed description.Typically, there will be more than one possible refinement of the architectural component that will satisfy the behavioral constraints.
- Choosing the best refinement is often a matter of trying to satisfy as many of the non-functional requirements of the system as possible. Thus the language used for the detailed design must allow some analysis of the design in order to assess its properties.

Coding and unit testing

- After coding,the component can be tested to verify that it performs correctly, according to some test criteria.

Integration and testing

- Enough components have been implemented and individually tested, they must be integrated as described in the architectural design. Further testing is done to ensure correct behavior and acceptable use of any shared resources. It is also possible at this time to perform some acceptance testing with the customers to ensure that the system meets their requirements.

Maintenance

- Category of maintenance, until such time as a new version of the product demands a total redesign or the product is phased out entirely. Maintenance involves the correction of errors in the system which are discovered after release and the revision of the system services to satisfy requirements that were not realized during previous development.

2.8.2 Validation and verification

- Throughout the life cycle, the design must be checked to ensure that it both satisfies the high-level requirements agreed with the customer and is also complete and internally consistent. These checks are referred to as validation and verification. Verification of a design will most often occur within a single life-cycle activity or between two adjacent activities.

- The detailed description will introduce more information in refining the general specification. The detailed design may also have to change the representations for the information and will almost certainly break up a single high-level operation into several low-level operations that can eventually be implemented.
- The changes to information and operations, the designer must show that the refined description is a legal one within its language (internal consistency) and that it describes all of the specified behavior of the high-level description (completeness) in a provably correct way (relative consistency).
- Validation of a design demonstrates that within the various activities the customer's requirements are satisfied. Validation is a much more subjective exercise than verification, mainly because the disparity between the language of the requirements and the language of the design forbids any objective form of proof.
- Languages with a mathematical foundation allow reasoning and proof in the objective sense. An argument based entirely within some mathematical language can be accepted or refuted based upon universally accepted measures. A proof can be entirely justified by the rules of the mathematical language, in which case it is considered a formal proof.
- More common is a rigorous proof, which is represented within some mathematical language but which relies on the understanding of the reader to accept its correctness without appeal to the full details of the argument, which could be provided but usually are not. The difference between formality and rigour is in the amount of detail the prover leaves out while still maintaining acceptance of the proof.
- Proofs that are for verification of a design can frequently occur within one language or between two languages which both have a precise mathematical semantics.
- Validation proofs this precludes the possibility of objective proof, rigorous or formal. Instead, there will always be a leap from the informal situations of the real world to any formal and structured development process. We refer to this inevitable disparity as the formality gap.

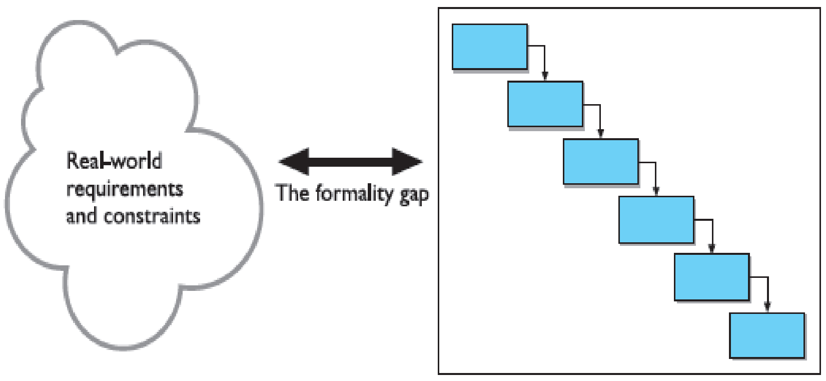


Fig 2.11: The formality gap between the real world and structured design

2.8.3 Interactive systems and the software life cycle

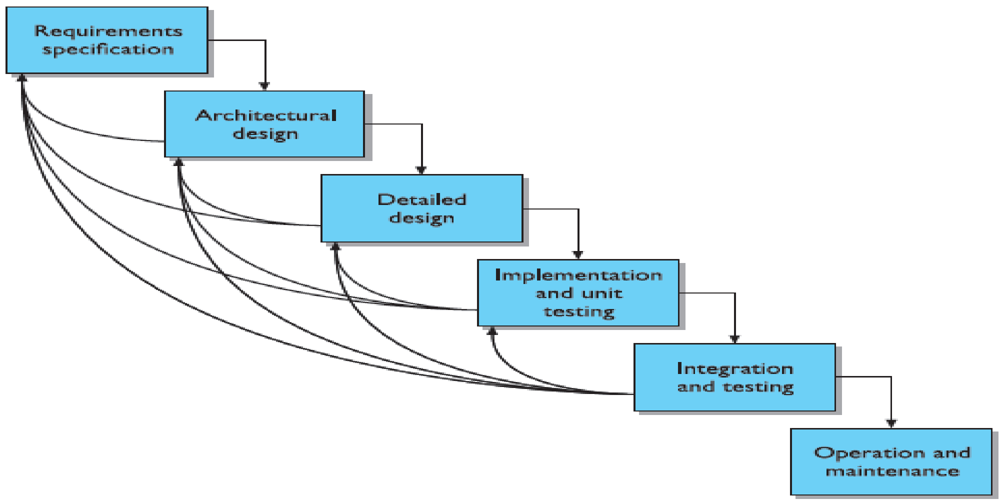

Requirements specification
Architectural design
Detailed design
Implementation and unit testing
Integration and testing
Operation and maintenance

Fig 2.12: Representing Iteration in the Waterfall Model

- The traditional software life cycle suits a principled approach to design; that is, if we know what it is we want to produce from the beginning, then we can structure our approach to design in order to attain the goal.
- The more serious claim we are making here is that all of the requirements for an interactive system cannot be determined from the start, and there are many convincing arguments to support this position. The result is that systems must be built and the interaction with users observed and evaluated in order to determine how to make them more usable.

2.9 USABILITY ENGINEERING

- One approach to user-centered design has been the introduction of explicit usability engineering goals into the design process, as suggested by Whiteside and colleagues at IBM and Digital Equipment Corporation [377] and by Nielsen at Bellcore [260,261]. Engineering depends on interpretation against a shared background of meaning, agreed goals and an understanding of how satisfactory completion will be judged.
- The emphasis for usability engineering is in knowing exactly what criteria will be used to judge a product for its usability. The ultimate test of a product's usability is based on measurements of users' experience with it. Therefore, since a user's direct experience with an interactive system is at the physical interface, focus on the actual user interface is understandable.
- The danger with this limited focus is that much of the work that is accomplished in interaction involves more than just the surface features of the systems used to perform that work. important features of usability engineering is the inclusion of a usability specification, forming part of the requirements Specification, that concentrates on features of the user–system interaction which contribute to the usability of the product. Various attributes of the system are suggested as gauges for testing the usability.

- The ultimate test of usability based on measurement of user experience. Usability specification consists of
  - usability attribute/principle
  - measuring concept

  - measuring method
  - now level/ worst case/ planned level/ best case

- For each attribute, six items are defined to form the usability specification of that attribute. Table provides an example of a usability specification for the design of a control panel for a video cassette recorder (VCR), based on the technique presented by Whiteside, Bennett and Holtzblatt [377].

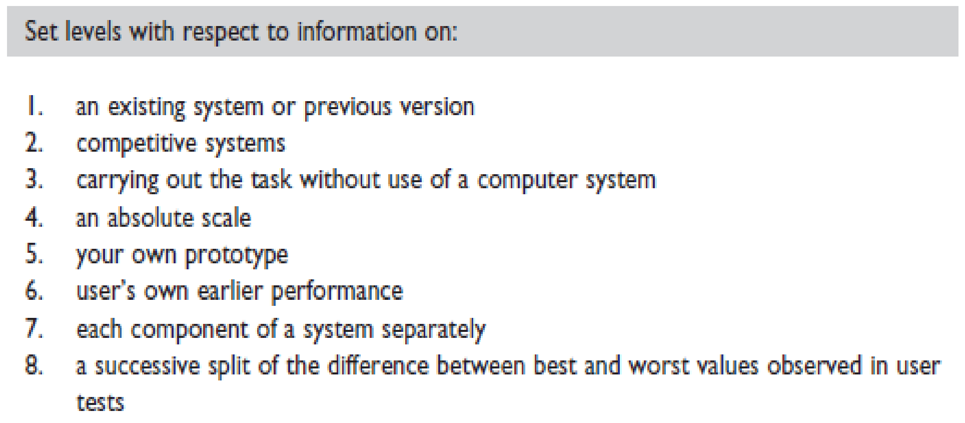
Attribute:	Backward recoverability
Measuring concept:	Undo an erroneous programming sequence
Measuring method:	Number of explicit user actions to undo current program
Now level:	No current product allows such an undo
Worst case:	As many actions as it takes to program in mistake
Planned level:	A maximum of two explicit user actions
Best case:	One explicit cancel action

Table 2.1: Sample Usability Specification for undo with a VCR

- The below tables and adapted from Whiteside, Bennett and Holtzblatt [377], provide a list of measurement criteria which can be used to determine the measuring method for a usability attribute and the possible ways to set the worst/best case and planned/ now level targets. Measurements such as those promoted by usability engineering are also called usability metrics.

1. Time to complete a task
2. Per cent of task completed
3. Per cent of task completed per unit time
4. Ratio of successes to failures
5. Time spent in errors
6. Per cent or number of errors
7. Per cent or number of competitors better than it
8. Number of commands used
9. Frequency of help and documentation use
10. Per cent of favorable/unfavorable user comments
11. Number of repetitions of failed commands
12. Number of runs of successes and of failures
13. Number of times interface misleads the user
14. Number of good and bad features recalled by users
15. Number of available commands not invoked
16. Number of regressive behaviors
17. Number of users preferring your system
18. Number of times users need to work around a problem
19. Number of times the user is disrupted from a work task
20. Number of times user loses control of the system
21. Number of times user expresses frustration or satisfaction

Table 2.2: Criteria by which measuring method can be determined



Set levels with respect to information on:

1. an existing system or previous version
2. competitive systems
3. carrying out the task without use of a computer system
4. an absolute scale
5. your own prototype
6. user's own earlier performance
7. each component of a system separately
8. a successive split of the difference between best and worst values observed in user tests

Table2.3: Possible ways to set measurement levels in a usability specification

2.9.1 Problems with usability engineering

- The major feature of usability engineering is the assertion of explicit usability metrics early on in the design process which can be used to judge a system once it is delivered. There is a very solid argument which points out that it is only through empirical approaches such as the use of usability metrics that we can reliably build more usable systems.
- The problem with usability metrics is that they rely on measurements of very specific user actions in very specific situations. When the designer knows what the actions and situation will be, then she can set goals for measured observations. However, at early stages of design, designers do not have this information

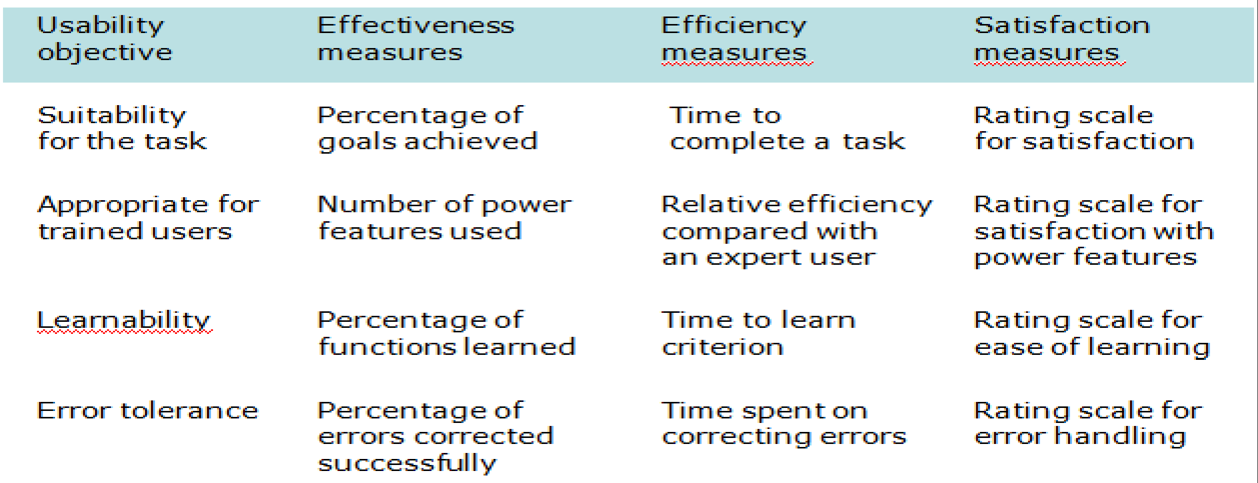
.

2.9.2 ISO usability standard 9241

It adopts traditional usability categories:

- effectiveness
  - can you achieve what you want to?
- efficiency
  - can you do it without wasting effort?
- satisfaction
  - do you enjoy the process?

Some metrics from ISO 9241



Usability objective	Effectiveness measures	Efficiency measures	Satisfaction measures
Suitability for the task	Percentage of goals achieved	Time to complete a task	Rating scale for satisfaction
Appropriate for trained users	Number of power features used	Relative efficiency compared with an expert user	Rating scale for satisfaction with power features
Learnability	Percentage of functions learned	Time to learn criterion	Rating scale for ease of learning
Error tolerance	Percentage of errors corrected successfully	Time spent on correcting errors	Rating scale for error handling

2.10 PROTOTYPING IN PRACTICE

- On the technical side, iterative design is described by the use of prototypes, artifacts that simulate or animate some but not all features of the intended system. There are three main approaches to prototyping:

    - throw-away
    - incremental
    - Evolutionary

a. Throw-away :The prototype is built and tested. The design knowledge gained from this exercise is used to build the final product, but the actual prototype is discarded. It depicts the procedure in using throw-away prototypes to arrive at a final requirements specification in order for the rest of the design process to proceed.

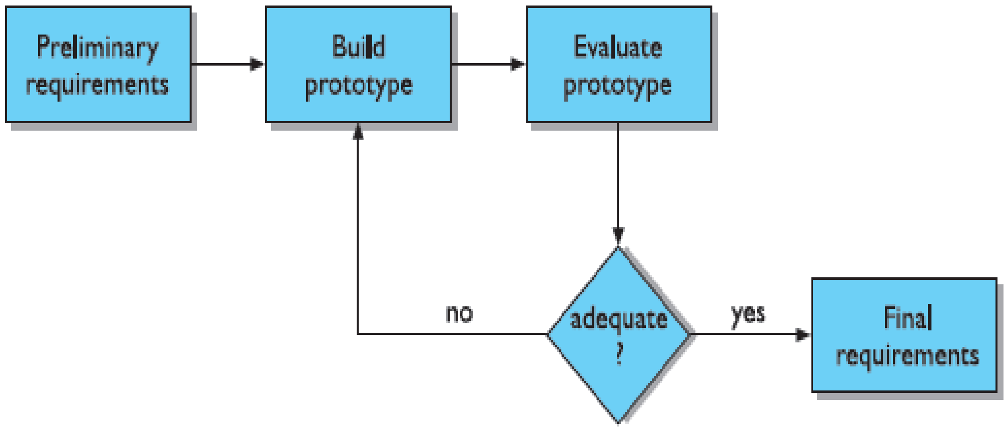


Fig 2.13: Throw-away prototyping with requirement specification

a. Incremental: The final product is built as separate components, one at a time. There is one overall design for the final system, but it is partitioned into independent and smaller components. The final product is then released as a series of products, each subsequent release including one more component. This is depicted in Figure.

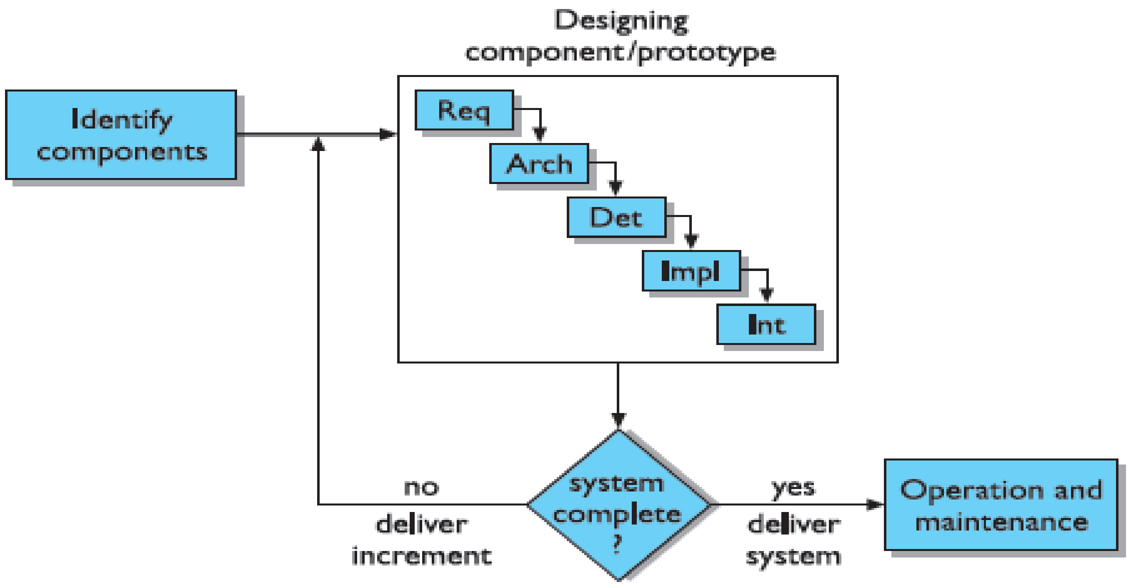


Fig 2.14: Incremental prototyping within the life cycle

b. Evolutionary: Here the prototype is not discarded and serves as the basis for the next iteration of design. In this case, the actual system is seen as evolving from a very limited initial version to its final release, as depicted in Figure.

- Evolutionary prototyping also fits in well with the modifications which must be made to the system that arise during the operation and maintenance activity in the life cycle.
- Prototypes differ according to the amount of functionality and performance they provide relative to the final product. An animation of requirements can involve no real functionality, or limited functionality to simulate only a small aspect of the interactive behavior for evaluative purposes.
- At the other extreme, full functionality can be provided at the expense of other performance characteristics, such as speed or error tolerance. Regardless of the level of functionality, the importance of a prototype lies in its projected realism.

- The prototype of an interactive system is used to test requirements by evaluating their impact with real users. An honest appraisal of the requirements of the final system can only be trusted if the evaluation conditions are similar to those anticipated for the actual operation. On the management side, there are several potential problems.

a. Time Building prototypes takes time and, if it is a throw-away prototype, it can be seen as precious time taken away from the real design task. So the value of prototyping is only appreciated if it is fast, hence the use of the term rapid prototyping.
b. Planning The project managers do not have the experience necessary for adequately planning and costing a design process which involves prototyping.
c. Non-functional features The most important features of a system will be non-functional ones, such as safety and reliability, and these are precisely the kinds of features which are sacrificed in developing a prototype.
d. Contracts The design process is often governed by contractual agreements between customer and designer which are affected by many of these managerial and technical issues. Prototypes and other implementations cannot form the basis for a legal contract, and so an iterative design process will still require documentation which serves as the binding agreement.

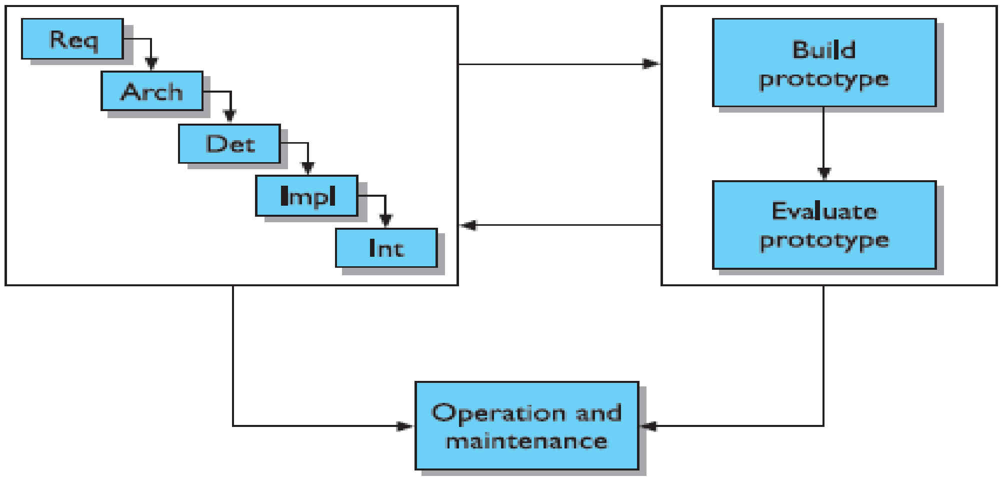


Fig 2.15: Evolutionary Prototyping throughout the life cycle

2.10.1 Techniques for prototyping

a. Storyboards: Storyboards do not require much in terms of computing power to construct; in fact, they can be mocked up without the aid of any computing resource. The origins of storyboards are in the film industry, where a series of panels roughly depicts snapshots from an intended film sequence in order to get the idea across about the eventual scene

b. Limited functionality simulations: The functionality must be built into the prototype to demonstrate the work that the application will accomplish. Storyboards and animation techniques are not sufficient for this purpose, as they cannot portray adequately the interactive aspects of the system.

c. High-level programming support: HyperTalk was an example of a special purpose high-level programming language which makes it easy for the designer to program certain features of an interactive system at the expense of other system features like speed of response or space efficiency. These high-level programming languages allow the programmer to abstract away from the hardware specifics and think in terms that are closer to the way the input and output devices are perceived as interaction devices.

2.10.2 Warning about iterative design

- The ideal model of iterative design, in which a rapid prototype is designed ,evaluated and modified until the best possible design is achieved in the given project time, is appealing with two problems.

  i) First, it is often the case that design decisions made at the very beginning of the prototyping process are wrong and, in practice, design inertia can be so great as never to overcome an initial bad decision

  ii) Second problem is slightly more subtle, and serious. If, in the process of evaluation, a potential usability problem is diagnosed, it is important to understand the reason for the problem and not just detect the symptom

2.11 DESIGN RATIONALE

- Designing any computer system, many decisions are made as the product goes from a set of vague customer requirements to a deliverable entity. Often it is difficult to recreate the reasons, or rationale, behind various design decisions.
- Design rationale is the information that explains why a computer system is the way it is, including its structural or architectural description and its functional or behavioral description .area of HCI, design rationale has been particularly important, again for several reasons:

❖ There is usually no single best design alternative. More often, the designer is faced with a set of trade-offs between different alternatives.

❖ Even if an optimal solution did exist for a given design decision, the space of alternatives is so vast that it is unlikely a designer would discover it. In this case, it is important that the designer indicates all alternatives that have been investigated.

❖ The usability of an interactive system is very dependent on the context of its use. The flashiest graphical interface is of no use if the end-user does not have access to a high-quality graphics display or a pointing device. Capturing the context in which a design decision is made will help later when new products are designed.

- There are some issues that distinguish the various techniques in terms of their usability within design itself. We can use these issues to sketch an informal rationale for design rationale. One issue is the degree to which the technique impinges on the design process.

- Another issue is the cost of using the technique, both in terms of creating the design rationale and in terms of accessing it once created. A related issue is the amount of computational power the design rationale provides and the level to which this is supported by automated tools.

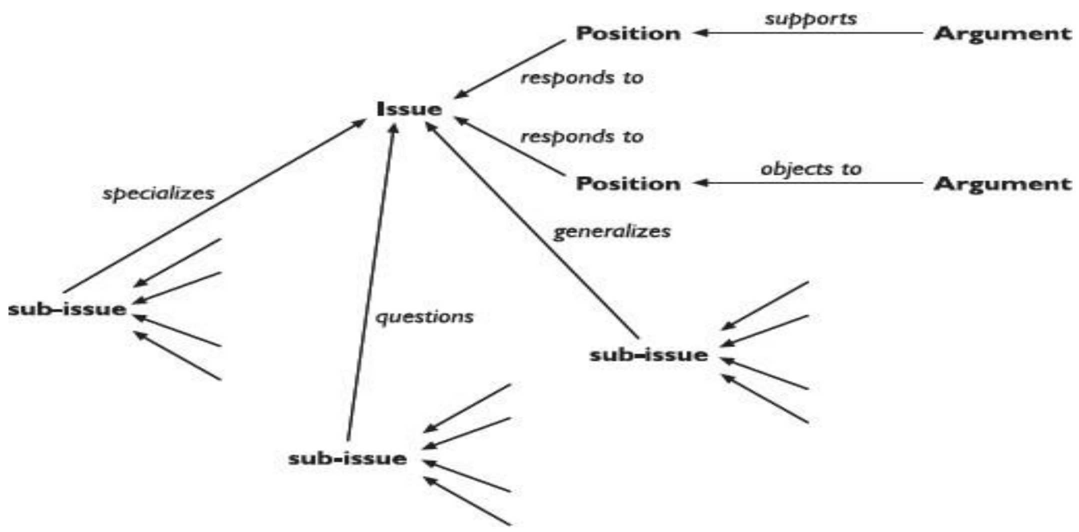
2.11.1 Process-oriented design rationale

- Much of the work on design rationale is based on Rittel's issue-based information system, or IBIS, a style for representing design and planning dialog developed in the 1970s [308]. In IBIS (pronounced 'ibbiss'), a hierarchical structure to a design rationale is created.

- A root issue is identified which represents the main problem or question that the argument is addressing. Various positions are put forth as potential resolutions for the root issue, and these are depicted as descendants in the IBIS hierarchy directly connected to the root issue. Each position is then supported or refuted by arguments, which modify the relationship between issue and position.

- The hierarchy grows as secondary issues are raised which modify the root issue in some way. Each of these secondary issues is in turn expanded by positions and arguments, further sub-issues, and so on.

- A graphical version of IBIS has been defined by Conklin and Yakemovic [77],called gIBIS (pronounced 'gibbiss'), which makes the structure of the design rationale more apparent visually in the form of a directed graph



which can be directly edited by the creator of the design rationale.

Fig 2.16: Structure of a gIBIS rationale

- The above Figure gives a representation of the gIBIS vocabulary. Issues, positions and arguments are nodes in the graph and the connections between them are labeled to clarify the relationship between adjacent nodes. So, for example, an issue can suggest further sub-issues, or a position can respond to an issue or an argument can support a position. The gIBIS structure can be supported by a hypertext tool to allow a designer to create and browse various parts of the design rationale.

2.11.2 Design space analysis

- MacLean and colleagues have proposed a more deliberative approach to design rationale which emphasizes a post hoc structuring of the space of design alternatives that have been considered in a design project. Their approach, embodied in the Questions, Options and Criteria (QOC) notation, is characterized as design space analysis.

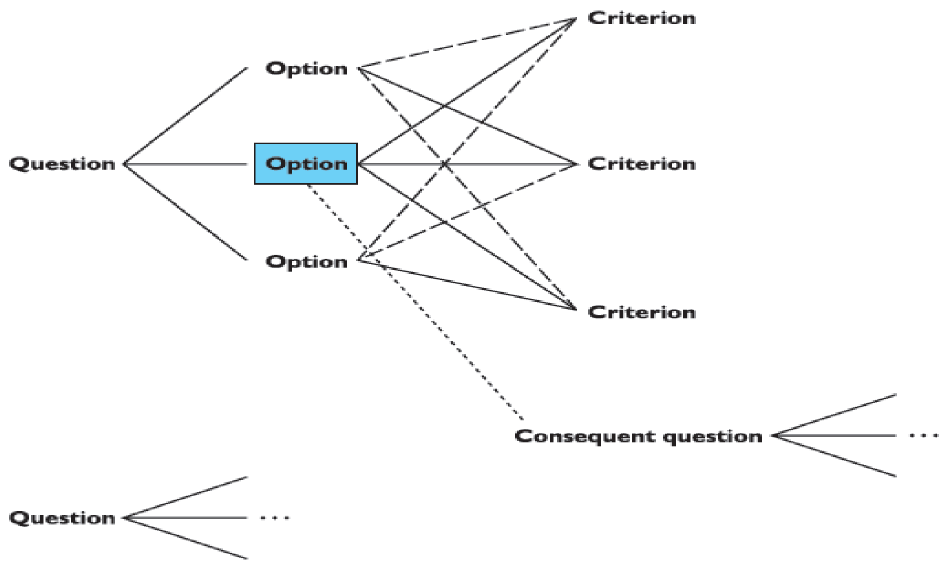


Fig 2.17: QOC notation

- The design space is initially structured by a set of questions representing the major issues of the design. Since design space analysis is structure oriented, it is not so important that the questions recorded are the actual questions asked during design meetings. Rather, these questions represent an agreed characterization of the issues raised based on reflection and understanding of the actual design activities.
- Questions in a design space analysis are therefore similar to issues in IBIS except in the way they are captured. Options provide alternative solutions to the question. They are assessed according to some criteria in order to determine the most favorable option. The key to an effective design space analysis using the QOC notation is deciding the right questions to use to structure the space and the correct criteria to judge the options.
- The initial questions raised must be sufficiently general that they cover a large enough portion of the possible design space, but specific enough that a range of options can be clearly identified. It can be difficult to decide the right set of criteria with which to assess the options. Another structure-oriented technique, called Decision Representation Language (DRL), developed by Lee and Lai, structures the design space in a similar fashion to QOC, though its language is somewhat larger and it has a formal semantics.
- The questions, options and criteria in DRL are given the names: decision problem, alternatives and goals. QOC assessments are represented in DRL by a more complex language for relating goals to alternatives. The sparse language in QOC used to assess an option relative to a criterion (positive or negative assessment only) is probably insufficient, but there is a trade-off involved in adopting a more complex vocabulary which may prove too difficult to use in practice. The advantage of the formal semantics of DRL is that the design rationale can be used as a computational mechanism to help manage the large volume of information.

2.11.3 Psychological design rationale

- The psychological design rationale has been introduced by Carroll and Rosson, and before we describe the application of the technique. When designing a new interactive system, the designers take into account the tasks that users currently perform and any new ones that they may want to perform.
- This task identification serves as part of the requirements for the new system, and can be done through empirical observation of how people perform their work currently and presented through informal language or a more formal task analysis language.
- Carroll refers to this real-life phenomenon as the task–artifact cycle. He provides a good example of this cycle through the evolution of the electronic spreadsheet. When the first electronic spreadsheet, VisiCalc, was marketed in the late 1970s, it was presented simply as an automated means of supporting tabular calculation, a task commonly used in the accounting world.
- The purpose of psychological design rationale is to support this natural task artifact cycle of design activity. The main emphasis is not to capture the designer's intention in building the artifact. Rather, psychological design rationale aims to make explicit the consequences of a design for the user, given an understanding of what tasks he intends to perform.
- The first step in the psychological design rationale is to identify the tasks that the proposed system will address and to characterize those tasks by questions that the user tries to answer in accomplishing them.

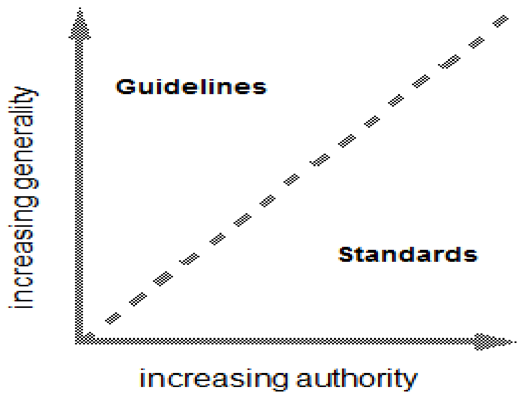
2.12 DESIGN RULES

- **Design**– the goal driven problem solving process
- **Design rules-** We require design rules, which are rules a designer can follow in order to increase the usability of the eventual software product. We can classify these rules along two dimensions, based on the rule's authority and generality.

- ➢ **authority** →mean an indication of whether or not the rule must be followed in design or whether it is only suggested.
- ➢ **Generality** → mean whether the rule can be applied to many design situations or whether it is focused on a more limited application situation.

- We will consider a number of different types of design rules.
  - a) **Principles** are abstract design rules, with high generality and low authority.
  - b) **Standards** are specific design rules, high in authority and limited in application.
  - c) **Guidelines** tend to be lower in authority and more general in application.
- Design rules for interactive systems can be supported by psychological, cognitive, ergonomic, sociological, economic or computational theory, which may or may not have roots in empirical evidence.
- Designers do not always have the relevant background in psychology, cognitive science, ergonomics, sociology, and business or computer science necessary to understand the consequences of those theories in the instance

of the design they are creating. Can make another rough distinction between principles, standards and guidelines.



- Principles are derived from knowledge of the psychological, computational and sociological aspects of the problem domains and are largely independent of the technology they depend to a much greater extent on a deeper understanding of the human element in the interaction. They can therefore be applied widely but are not so useful for specific design advice.
- Guidelines are less abstract and often more technology oriented, but as they are also general, it is important for a designer to know what theoretical evidence there is to support them. A designer will have less of a need to know the underlying theory for applying a standard.
- Design rules are mechanisms for restricting the space of design options, preventing a designer from pursuing design options that would be likely to lead to an unusable system. Thus, design rules would be most effective if they could be adopted in the earliest stages of the life cycle, such as in requirements specification and architectural design, when the space of possible designs is still very large

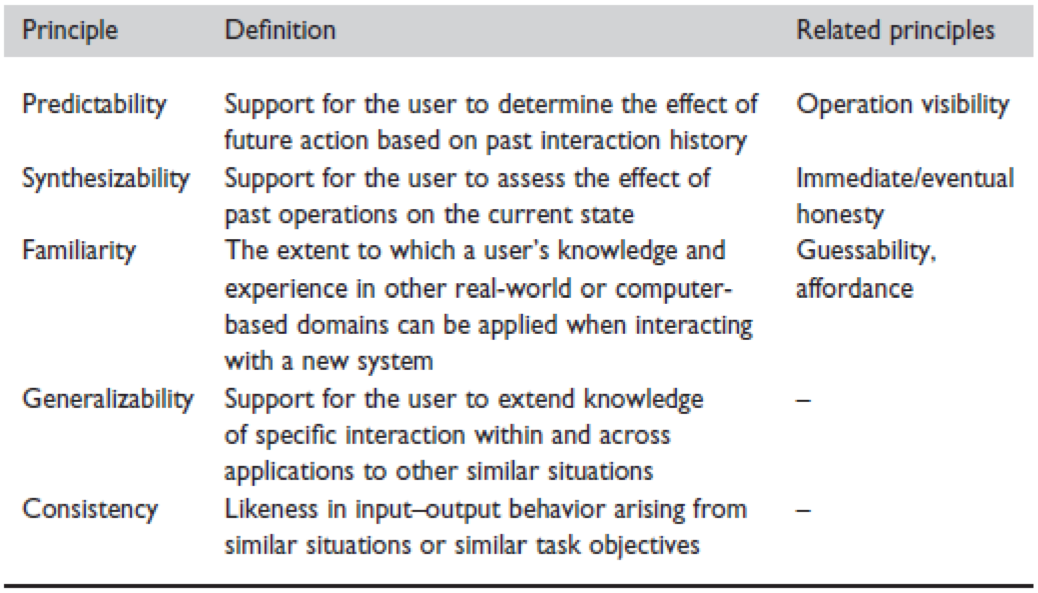
2.13 PRINCIPLES

- The most abstract design rules are general principles, which can be applied to the design of an interactive system in order to promote its usability. Derivation of principles for interaction has usually arisen out of a need to explain why a paradigm is successful and when it might not be.
- Principles can provide the repeatability which paradigms in themselves cannot provide. In this section we present a collection of usability principles.The principles we present are first divided into three main categories:

1. **Learnability** – the ease with which new users can begin effective interaction and achieve maximal performance.
2. **Flexibility** – the multiplicity of ways in which the user and system exchange information.
3. **Robustness** – the level of support provided to the user in determining successful achievement and assessment of goals.

2.13.1 Learnability

- Concerns the features of the interactive system that allow novice users to understand how to use it initially and then how to attain a maximal level of performance



Principle	Definition	Related principles
Predictability	Support for the user to determine the effect of future action based on past interaction history	Operation visibility
Synthesizability	Support for the user to assess the effect of past operations on the current state	Immediate/eventual honesty
Familiarity	The extent to which a user's knowledge and experience in other real-world or computer-based domains can be applied when interacting with a new system	Guessability, affordance
Generalizability	Support for the user to extend knowledge of specific interaction within and across applications to other similar situations	–
Consistency	Likeness in input–output behavior arising from similar situations or similar task objectives	–

Table 2.4: Summary of principles affecting learnability

a. Predictability: Predictability of an interactive system is distinguished from deterministic behavior of the computer system alone. Most computer systems are ultimately deterministic machines, so that given the state at any one point in time and the operation which is to be performed at that time, there is only one possible state that can result..

b. Synthesizability: It is the ability of the user to assess the effect of past operations on the current state. When an operation changes some aspect of the internal state, it is important that the change is seen by the user. The principle of honesty relates to the ability of the user interface to provide an observable and informative account of such change.

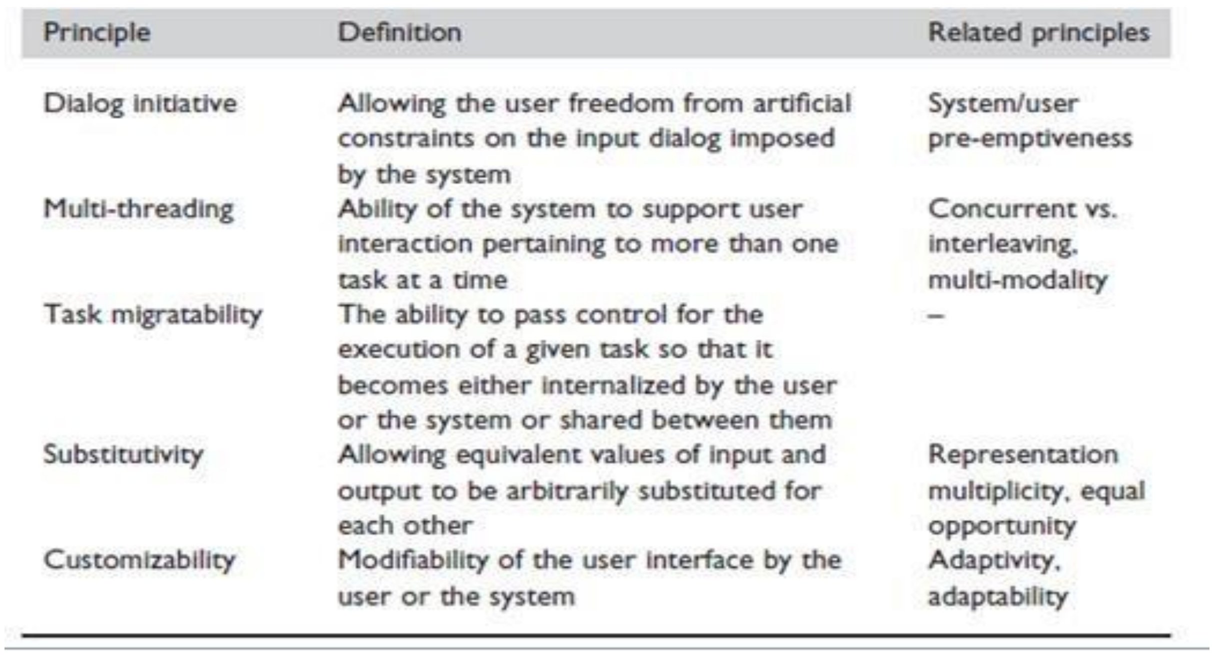
c. Familiarity: The familiarity of an interactive system measures the correlation between the user's existing knowledge and the knowledge required for effective interaction. Familiarity has to do with a user's first impression of the system.

d. Generalizability: The generalizability of an interactive system supports this activity, leading to a more complete predictive model of the system for the user. We can apply generalization to situations in which the user wants to apply knowledge that helps achieve one particular goal to another situation where the goal is in some way similar. Generalizability can be seen as a form of consistency. Generalization can occur within a single application or across a variety of applications.

e. Consistency: Consistency relates to the likeness in behavior arising from similar situations or similar task objectives. Consistency is probably the most widely mentioned principle Another consequence of consistency having to be defined with respect to some other feature of the interaction is that many other principles can be 'reduced' to qualified instances of consistency.

2.13.2 Flexibility

- Flexibility refers to the multiplicity of ways in which the end-user and the system exchange information.



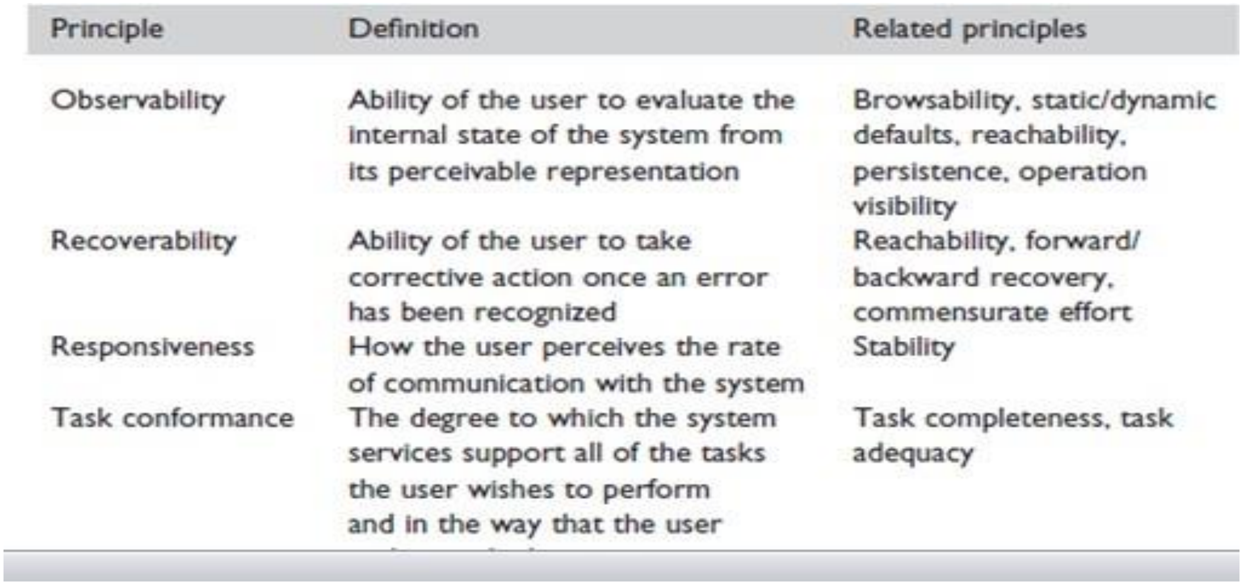
Principle	Definition	Related principles
Dialog initiative	Allowing the user freedom from artificial constraints on the input dialog imposed by the system	System/user pre-emptiveness
Multi-threading	Ability of the system to support user interaction pertaining to more than one task at a time	Concurrent vs. interleaving, multi-modality
Task migratability	The ability to pass control for the execution of a given task so that it becomes either internalized by the user or the system or shared between them	–
Substitutivity	Allowing equivalent values of input and output to be arbitrarily substituted for each other	Representation multiplicity, equal opportunity
Customizability	Modifiability of the user interface by the user or the system	Adaptivity, adaptability

Table 2.5: Summary of principles affecting flexibility

a. Dialog initiative: The system can initiate all dialogs, in which case the user simply responds to requests for information. We call this type of dialog system pre-emptive.the user may be entirely free to initiate any action towards the system, in which case the dialog is user pre-emptive. The system may control the dialog to the extent that it prohibits the user from initiating any other desired communication concerning the current task or some other task the user would like to perform.
b. Multi-threading: A thread of a dialog is a coherent subset of that dialog. In the user–system dialog, we can consider a thread to be that part of the dialog that relates to a given user task. Multi-threading of the user–system dialog allows for interaction to support more than one task at a time.
c. Task migratability: Concerns the transfer of control for execution of tasks between system and user. It should be possible for the user or system to pass the control of a task over to the other or promote the task from a completely internalized one to a shared and cooperative venture. Hence, a task that is internal to one can become internal to the other or shared between the two partners.
d. Substitutivity: It requires that equivalent values can be substituted for each other. For example, in considering the form of an input expression to determine the margin for a letter, you may want to enter the value in either inches or centimeters.
e. Customizability: Customizability is the modifiability of the user interface by the user or the system. From the system side, we are not concerned with modifications that would be attended to by a programmer actually changing the system and its interface during system maintenance.

2.13.3 Robustness

- The robustness of that interaction covers features that support the successful achievement and assessment of the goals. Here, we describe principles that support robustness.



Principle	Definition	Related principles
Observability	Ability of the user to evaluate the internal state of the system from its perceivable representation	Browsability, static/dynamic defaults, reachability, persistence, operation visibility
Recoverability	Ability of the user to take corrective action once an error has been recognized	Reachability, forward/ backward recovery, commensurate effort
Responsiveness	How the user perceives the rate of communication with the system	Stability
Task conformance	The degree to which the system services support all of the tasks the user wishes to perform and in the way that the user	Task completeness, task adequacy

Table 2.6: Summary of principles affecting Robustness

a. Observability: It allows the user to evaluate the internal state of the system by means of its perceivable representation at the interface. The evaluation allows the user to compare the current observed state with his intention within the task–action plan, possibly leading to a plan revision. It can be discussed through five other principles: browsability, defaults, reachability, persistence and operation visibility.

Browsability allows the user to explore the current internal state of the system via the limited view provided at the interface.

Defaults It also reduces the number of physical actions necessary to input a value. Thus, providing default values is a kind of error prevention mechanism. There are two kinds of default values: static and dynamic. Static defaults do not evolve with the session

Reachability refers to the possibility of navigation through the observable system states. There are various levels of reachability that can be given precise mathematical definitions , but the main notion is whether the user can navigate from any given state to any other state.

Persistence deals with the duration of the effect of a communication act and the ability of the user to make use of that effect. The effect of vocal communication does not persist except in the memory of the receiver. Visual communication, on the other hand, can remain as an object which the user can subsequently manipulate long after the act of presentation.

b. Recoverability: It is the ability to reach a desired goal after recognition of some error in a previous interaction. There are two directions in which recovery can occur, forward or backward.

- ❖ Forward error recovery involves the acceptance of the current state and negotiation from that state towards the desired state. Forward error recovery may be the only possibility for recovery if the effects of interaction are not revocable.
- ❖ Backward error recovery is an attempt to undo the effects of previous interaction in order to return to a prior state before proceeding. In a text editor, a mistyped keystroke might wipe out a large section of text which you would want to retrieve by an equally simple undo button

c. Responsiveness: It measures the rate of communication between the system and the user. Response time is generally defined as the duration of time needed by the system to express state changes to the user. Significant as absolute response time is response time stability. Response time stability covers the invariance of the duration for identical or similar computational resources.
d. Task conformance: Task completeness addresses the coverage issue and task adequacy addresses the user's understanding of the tasks. Task completeness refers to the level to which the system services can be mapped onto all of the user tasks.

2.15 Standards

- set by national or international bodies to ensure compliance by a large community of designers standards require sound underlying theory and slowly changing technology
- hardware standards more common than software high authority and low level of detail
- ISO 9241 defines usability as effectiveness, efficiency and satisfaction with which users accomplish tasks (refer pg no 23)

2.16 GUIDELINES

- It concern in examining the wealth of available guidelines is in determining their applicability to the various stages of design. The guidelines can also be automated to some extent, providing a direct means for translating detailed design specifications into actual implementation. There are a vast amount of published guidelines for interactive system design (they are frequently referred to as guidelines for user interface design). The basic categories of the Smith and Mosier guidelines are:
    1. Data Entry
    2. Data Display
    3. Sequence Control
    4. User Guidance
    5. Data Transmission
    6. Data Protection
- A major concern for all of the general guidelines is the subject of dialog styles,which in the context of these guidelines pertains to the means by which the user communicates input to the system, including how the system presents the communication device.
- Smith and Mosier identify eight different dialog styles and Mayhew identifies seven. The only real difference is the absence of query languages in Mayhew's list, but we can consider a query language as a special case of a command language.
- In moving from abstract guidelines to more specific and automated ones, it is necessary to introduce assumptions about the computer platform on which the interactive system is designed. So, for example, in Apple's Human Interface Guidelines: the Apple Desktop Interface, there is a clear distinction between the abstract guidelines (or principles), independent of the specific Macintosh hardware and software, and the concrete guidelines, which assume them.

2.17 RULES

- There are many sets of heuristics, but the most well used are
  - Nielsen's ten heuristics,
  - Shneiderman's eight golden rules
  - Norman's seven principles

2.17.1 Shneiderman's Eight Golden Rules of Interface Design

- Shneiderman's eight golden rules provide a convenient and succinct summary of the key principles of interface design. They are intended to be used during design but can also be applied, like Nielsen's heuristics, to the evaluation of systems. Notice how they relate to the abstract principles discussed earlier.

1. Strive for consistency in action sequences, layout, terminology, command use and so on.
2. Enable frequent users to use shortcuts, such as abbreviations, special key sequences and macros, to perform regular, familiar actions more quickly.
3. Offer informative feedback for every user action, at a level appropriate to the magnitude of the action.
4. Design dialogs to yield closure so that the user knows when they have completed a task.
5. Offer error prevention and simple error handling so that, ideally, users are prevented from making mistakes and, if they do, they are offered clear and informative instructions to enable them to recover.
6. Permit easy reversal of actions in order to relieve anxiety and encourage exploration, since the user knows that he can always return to the previous state.
7. Support internal locus of control so that the user is in control of the system, which responds to his actions.
8. Reduce short-term memory load by keeping displays simple, consolidating multiple page displays and providing time for learning action sequences.

- These rules provide a useful shorthand for the more detailed sets of principles described earlier. Like those principles, they are not applicable to every eventuality and need to be interpreted for each new situation. However, they are broadly useful and their application will only help most design projects.

2.17.2 Norman's Seven Principles for Transforming Difficult Tasks into Simple Ones

- We discussed Norman's execution–evaluation cycle, in which he elaborates the seven stages of action. We using the following seven principles:

1. Use both knowledge in the world and knowledge in the head. People work better when the knowledge they need to do a task is available externally – either explicitly or through the constraints imposed by the environment. But experts also need to be able to internalize regular tasks to increase their efficiency. So systems should provide the necessary knowledge within the environment and their operation should be transparent to support the user in building an appropriate mental model of what is going on.
2. Simplify the structure of tasks. Tasks need to be simple in order to avoid complex problem solving and excessive memory load. There are a number of ways to simplify the structure of tasks. One is to provide mental aids to help the user keep track of stages in a more complex task. Another is to use technology to provide the user with more information about the task and better feedback. A

third approach is to automate the task or part of it, as long as this does not detract from the user's experience. The final approach to simplification is to change the nature of the task so that it becomes something more simple. In all of this, it is important not to take control away from the user.

3. Make things visible: bridge the gulfs of execution and evaluation. The interface should make clear what the system can do and how this is achieved, and should enable the user to see clearly the effect of their actions on the system.

4. Get the mappings right. User intentions should map clearly onto system controls. User actions should map clearly onto system events. So it should be clear what does what and by how much. Controls, sliders and dials should reflect the task so a small movement has a small effect and a large movement a large effect.

5. Exploit the power of constraints, both natural and artificial. Constraints are things in the world that make it impossible to do anything but the correct action in the correct way. A simple example is a jigsaw puzzle, where the pieces only fit together in one way. Here the physical constraints of the design guide the user to complete the task.

6. Design for error. To err is human, so anticipate the errors the user could make and design recovery into the system.

7. When all else fails, standardize. If there are no natural mappings then arbitrary mappings should be standardized so that users only have to learn them once. It is this standardization principle that enables drivers to get into a new car and drive it with very little difficulty – key controls are standardized. Occasionally one might switch on the indicator lights instead of the windscreen wipers, but the critical controls (accelerator, brake, clutch, steering) are always the same.

2.17.3 Nielsen's ten heuristics are:

1. **Visibility of system status** Always keep users informed about what is going on, through appropriate feedback within reasonable time. For example, if a system operation will take some time, give an indication of how long and how much is complete.
2. **Match between system and the real world** The system should speak the user's language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in natural and logical order.
3. **User control and freedom** Users often choose system functions by mistake and need a clearly marked 'emergency exit' to leave the unwanted state without having to go through an extended dialog. Support undo and redo.
4. **Consistency and standards** Users should not have to wonder whether words, situations or actions mean the same thing in different contexts. Follow platform conventions and accepted standards.
5. **Error prevention** Make it difficult to make errors. Even better than good error messages is a careful design that prevents a problem from occurring in the first place.
6. **Recognition rather than recall** Make objects, actions and options visible. The user should not have to remember information from one part of the dialog to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
7. **Flexibility and efficiency of use** Allow users to tailor frequent actions.

Accelerators – unseen by the novice user – may often speed up the interaction for the expert user to such an extent that the system can cater to both inexperienced and experienced users.
8. **Aesthetic and minimalist design** Dialogs should not contain information that is irrelevant or rarely needed. Every extra unit of information in a dialog competes with the relevant units of information and diminishes their relative visibility.
9. **Help users recognize**, **diagnose and recover from errors** Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
10. **Help and documentation** Few systems can be used with no instructions so it may be necessary to provide help and documentation. Any such information should be easy to search, focussed on the user's task, list concrete steps to be carried out, and not be too large.

HCI design patterns

An approach to reusing knowledge about successful design solutions

- Originated in architecture: Alexander
- A pattern is an invariant solution to a recurrent problem within a specific context.
- Examples
    - Light on Two Sides of Every Room (architecture)
    - Go back to a safe place (HCI)
- Patterns do not exist in isolation but are linked to other patterns in *languages* which enable complete designs to be generated
- Characteristics of patterns
    - capture design practice not theory
    - capture the essential common properties of good examples of design
    - represent design knowledge at varying levels: social, organisational, conceptual, detailed
    - are intuitive and readable and can therefore be used for communication between all stakeholders
    - a pattern language should be generative and assist in the development of complete designs.

2.18 EVALUATION TECHNIQUES – UNIVERSAL DESIGN

Evaluation

- tests usability and functionality of system
- occurs in laboratory, field and/or in collaboration with users
- evaluates both design and implementation
- should be considered at all stages in the design life cycle

- Evaluation should not be thought of as a single phase in the design process (still less as an activity tacked on the end of the process if time permits). Ideally, evaluation should occur throughout the design life cycle, with the results of the evaluation feeding back into modifications to the design.
- It is not usually possible to perform extensive experimental testing continuously throughout the design, but analytic and informal techniques can and should be used close link between evaluation and the principles and prototyping techniques.

- Such techniques help to ensure that the design is assessed continually. This has the advantage that problems can be ironed out before considerable effort and resources have been expended on the implementation itself: it is much easier to change a design in the early stages of development than in the later stages.
- Evaluation has three main goals:
    a) To assess the extent and accessibility of the system's functionality
    b) To assess users' experience of the interaction
    c) To identify any specific problems with the system.

Evaluation Techniques

1. Evaluating Designs through Expert Analysis
2. Evaluating Designs through user participation
3. Evaluating Implementations
    - Empirical or experimental methods
    - Observational methods
    - Query techniques
    - Methods that use physiological monitoring

2.18.1 Evaluation Through Expert Analysis

We will consider four approaches to expert analysis:

- Cognitive Walkthrough
- Heuristic Evaluation
- Model based Evaluation
- Review-based evaluation

a. Cognitive walkthrough

Cognitive -Psychological processes involved in acquisition and understanding of knowledge, formation of beliefs and attitudes, and decision making and problem solving.

Cognitive Walkthrough-usability evaluation method in which one or more evaluators work through a series of tasks and ask a set of questions from the perspective of the user. The focus of the cognitive walkthrough is on understanding the system's learnability for new or infrequent users

- Cognitive walkthrough was originally proposed and later revised by Polson and colleagues as an attempt to introduce psychological theory into the informal and subjective walkthrough technique. The origin of the cognitive walkthrough approach to evaluation is the code walkthrough familiar in software engineering.
- Walkthroughs require a detailed review of a sequence of actions. In the code walkthrough, the sequence represents a segment of the program code that is stepped through by the reviewers to check certain characteristics. To do a walkthrough (the term walkthrough from now on refers to the cognitive walkthrough, and not to any other kind of walkthrough), you need four things:

1. A specification or prototype of the system. It doesn't have to be complete, but it should be fairly detailed. Details such as the location and wording for a menu can make a big difference.
2. A description of the task the user is to perform on the system. This should be a representative task that most users will want to do.

3. A complete, written list of the actions needed to complete the task with the proposed system.
4. An indication of who the users are and what kind of experience and knowledge the evaluators can assume about them.

b. Heuristic evaluation
Heuristic -enabling a person to discover or learn something for themselves
Heuristic Evaluation-helps to identify usability problems in the user interface (UI) design. It specifically involves evaluators examining the interface and judging its compliance with recognized usability principles

- Example heuristics
  - system behaviour is predictable
  - system behaviour is consistent
  - feedback is provided

- Heuristic evaluation, developed by Jakob Nielsen and Rolf Molich, is a method for structuring the critique of a system using a set of relatively simple and general heuristics. Heuristic evaluation can be performed on a design specification so it is useful for evaluating early design. But it can also be used on prototypes, storyboards and fully functioning systems. It is therefore a flexible, relatively cheap approach. Hence it is often considered a discount usability technique. Nielsen's ten heuristics are: **(pg no 36)**
- The evaluator assesses the severity of each usability problem, based on four factors:
  - How common is the problem
  - How easy is it for the user to overcome
  - Will it be a one-off problem or a persistent one
  - How seriously will the problem be perceived

These can be combined into an overall severity rating on a scale of 0–4:
0 = I don't agree that this is a usability problem at all
1 = Cosmetic problem only: need not be fixed unless extra time is available on project
2 = Minor usability problem: fixing this should be given low priority
3 = Major usability problem: important to fix, so should be given high priority
4 = Usability catastrophe: imperative to fix this before product can be released

Model Based Evaluation
- This evaluation is done with the use of models
- The GOMS (goals, operators, methods and selection) model predicts user performance with a particular interface and can be used to filter particular design options
- Dialog models can also be used to evaluate dialog sequences for problems, such as unreachable states, complexity.
- Models such as state transition networks are useful for evaluating dialog designs prior to implementation

Review-based evaluation
- Results from the literature used to support or refute parts of design.
- Care needed to ensure results are transferable to new design.

- *Expert review: expertise in the area is required to ensure that correct assumptions* are made

2.18.2 Evaluating through user Participation

Two distinct evaluation styles

- Laboratory studies
- Field Studies

Laboratory studies

In this approach, users are taken out of their normal work environment to take part in controlled tests, often in a specialist usability laboratory

Advantages:

- specialist equipment available
- uninterrupted environment

Disadvantages:

- lack of context
- difficult to observe several users cooperating

Appropriate

- if system location is dangerous or impractical for constrained single user systems to allow controlled manipulation of use

Field Studies

This approach takes the designer or evaluator out into the user's work environment in order to observe the system in action

Advantages:

- natural environment
- context retained (though observation may alter it)
- longitudinal studies possible

Disadvantages:

- distractions
- noise

Appropriate

- where context is crucial for longitudinal studies

2.18.3.Evaluating Implementations

It has 4 types.

a) Empirical or experimental methods
b) Observational methods
c) Query techniques
d) Methods that use physiological monitoring

a) Empirical or experimental methods

- Controlled evaluation of specific aspects of interactive behaviour
- Evaluator chooses hypothesis to be tested
- Number of experimental conditions are considered which differ only in the value of some controlled variable.
- Changes in behavioural measure are attributed to different conditions

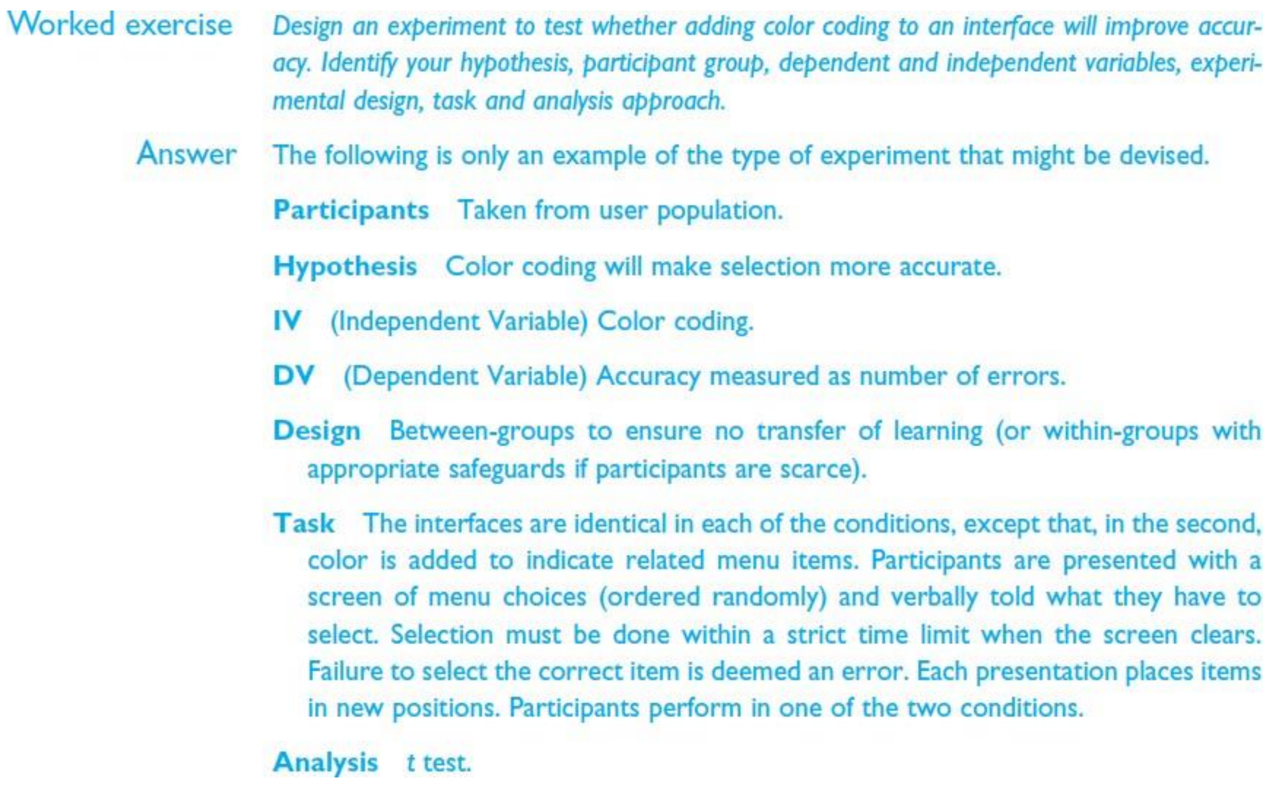
  Experimental factors
- Participants

- participants should be chosen to match the expected user population as closely as possible
- Sample Size chosen

• Variables
  - things to modify and measure
  - **Independent variable (IV)**
  - characteristic changed to produce different conditions
  - e.g. interface style, number of menu items
  - **Dependent variable (DV)**
  - characteristics measured in the experiment
  - e.g. time taken, number of errors

• Hypothesis
  - Prediction of the outcome of an experiment
  - To show that this prediction is correct

• Experimental design
  - First phase is to choose the hypothesis: to decide exactly what it is you are trying to demonstrate.
  - Next step is to decide on the *experimental method.*
  - Two main methods:
    - ***between-subjects-***each participant is assigned to a different condition*. A*t least two conditions: the experimental condition and the control
    - ***within-subjects* -** each user performs under each different condition



Worked exercise *Design an experiment to test whether adding color coding to an interface will improve accuracy. Identify your hypothesis, participant group, dependent and independent variables, experimental design, task and analysis approach.*

Answer The following is only an example of the type of experiment that might be devised.

Participants Taken from user population.

Hypothesis Color coding will make selection more accurate.

IV (Independent Variable) Color coding.

DV (Dependent Variable) Accuracy measured as number of errors.

Design Between-groups to ensure no transfer of learning (or within-groups with appropriate safeguards if participants are scarce).

Task The interfaces are identical in each of the conditions, except that, in the second, color is added to indicate related menu items. Participants are presented with a screen of menu choices (ordered randomly) and verbally told what they have to select. Selection must be done within a strict time limit when the screen clears. Failure to select the correct item is deemed an error. Each presentation places items in new positions. Participants perform in one of the two conditions.

Analysis *t* test.

b) Observational methods

It has the following methods

- Think Aloud & Cooperative evaluation
- Protocol analysis
- Automated analysis
- Post-task walkthroughs

Think Aloud &

- User observed performing task
- User asked to describe what he is doing and why, what he thinks is happening etc.
- Advantages
  - simplicity - requires little expertise
  - can provide useful insight
  - can show how system is actually use
- Disadvantages
  - subjective
  - selective
  - act of describing may alter task performance

Cooperative evaluation

- variation on think aloud
- user collaborates in evaluation
- both user and evaluator can ask each other questions throughout
- Additional advantages
  - less constrained and easier to use
  - user is encouraged to criticize system
  - clarification possible

Protocol analysis

- Paper and pencil – cheap, limited to writing speed
- Audio – good for think aloud, difficult to match with other protocols
- Video – accurate and realistic, needs special equipment
- Computer logging – automatic and modest, large amounts of data difficult to analyze
- User notebooks – coarse and subjective, useful insights, good for longitudinal studies
- Mixed use in practice.
- audio/video transcription difficult and requires skill.
- Some automatic support tools available

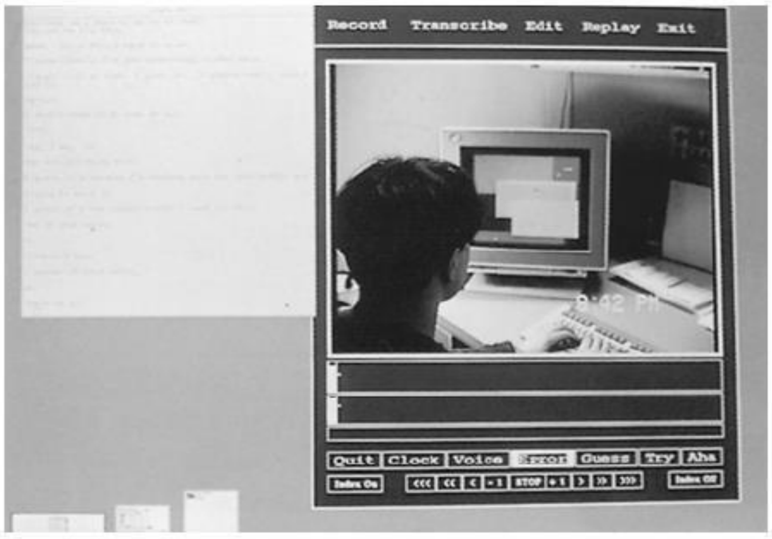
Automated analysis – EVA(*Experimental Video Annotator*)

- Workplace project - multimedia workstation

with a direct link to a video recorder

- Post task walkthrough
  - user reacts on action after the event
  - used to fill in intention
- Advantages
  - analyst has time to focus on relevant incidents
  - avoid excessive interruption of task
- Disadvantages

- lack of freshness
- may be post-hoc interpretation of events



EVA: an automatic protocol analysis tool. Source: Wendy I

Post-task walkthroughs

- The transcript, whether written or recorded, is replayed to the participant who is invited to comment, or is directly questioned by the analyst.
- In some circumstances the participant cannot be expected to talk during the actual observation.
- Necessary in cases where think aloud is not possible

c) Query techniques (Interviews , Questionnaires)

Interviews

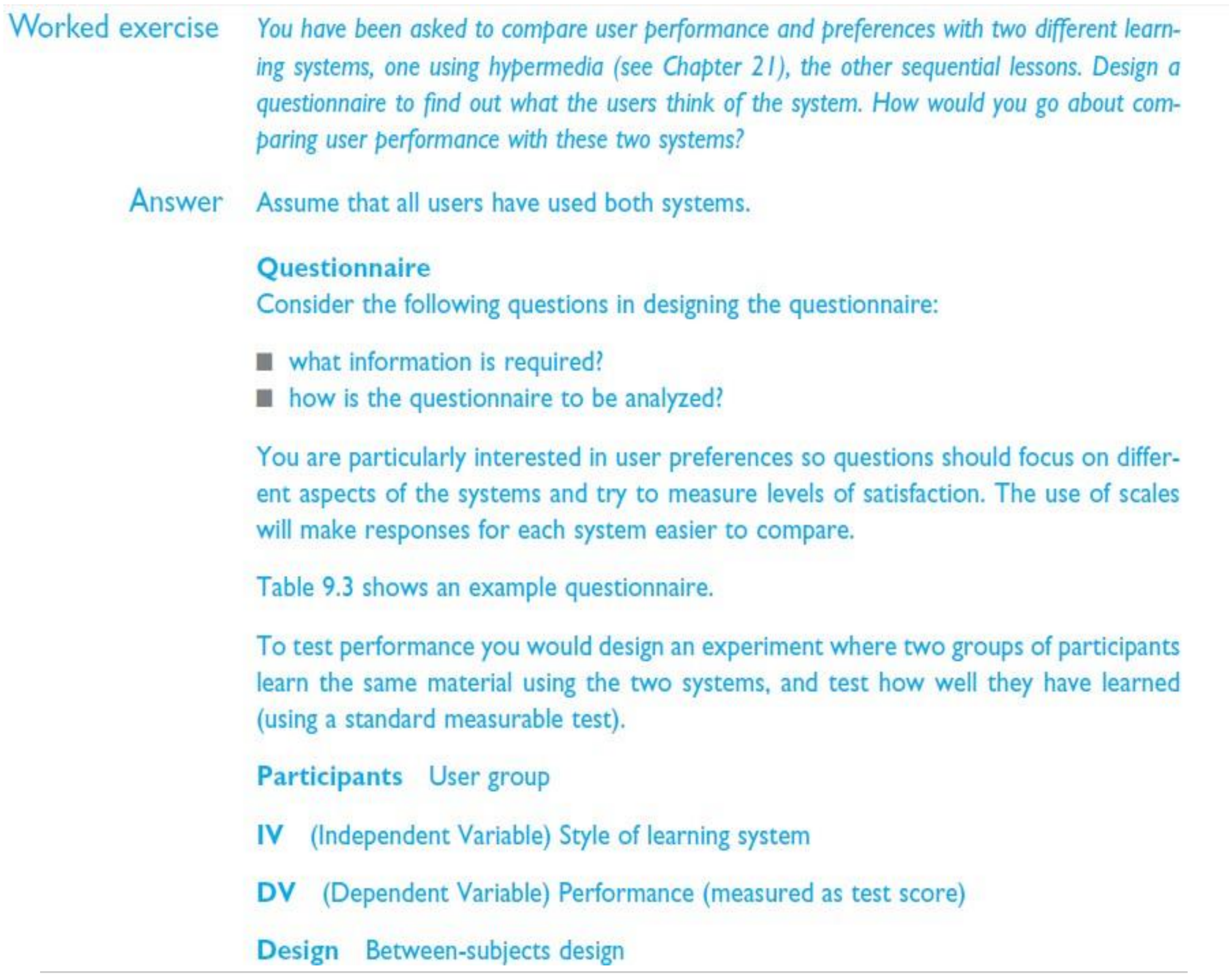
- Analyst questions user on one-to -one basis usually based on prepared questions
- Informal, subjective and relatively cheap
- Advantages
  - can be varied to suit context
  - issues can be explored more fully
  - can elicit user views and identify unanticipated problems
- Disadvantages
  - very subjective
  - time consuming

Questionnaires

- Set of fixed questions given to users
- Advantages
  - quick and reaches large user group
  - can be analyzed more rigorously
- Disadvantages
  - less flexible
  - less probing

- Need careful design
  - what information is required?

  - how are answers to be analyzed?
- Styles of question
  - general
  - open-ended
  - scalar
  - multi-choice
  - ranked



Worked exercise *You have been asked to compare user performance and preferences with two different learning systems, one using hypermedia (see Chapter 21), the other sequential lessons. Design a questionnaire to find out what the users think of the system. How would you go about comparing user performance with these two systems?*

Answer Assume that all users have used both systems.

Questionnaire
Consider the following questions in designing the questionnaire:

- what information is required?
- how is the questionnaire to be analyzed?

You are particularly interested in user preferences so questions should focus on different aspects of the systems and try to measure levels of satisfaction. The use of scales will make responses for each system easier to compare.

Table 9.3 shows an example questionnaire.

To test performance you would design an experiment where two groups of participants learn the same material using the two systems, and test how well they have learned (using a standard measurable test).

Participants User group

IV (Independent Variable) Style of learning system

DV (Dependent Variable) Performance (measured as test score)

Design Between-subjects design

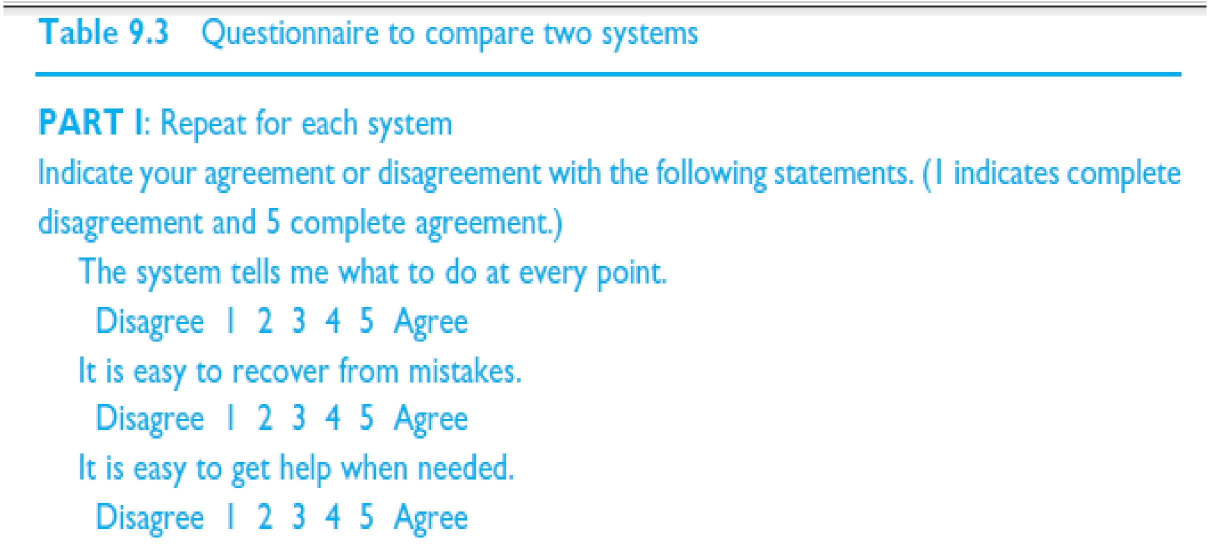


Table 9.3 Questionnaire to compare two systems

PART I: Repeat for each system
Indicate your agreement or disagreement with the following statements. (1 indicates complete disagreement and 5 complete agreement.)

The system tells me what to do at every point.
Disagree 1 2 3 4 5 Agree

It is easy to recover from mistakes.
Disagree 1 2 3 4 5 Agree

It is easy to get help when needed.
Disagree 1 2 3 4 5 Agree

I always know what the system is doing.
Disagree 1 2 3 4 5 Agree
I always know where I am in the training material.
Disagree 1 2 3 4 5 Agree
I have learned the material well using the system.
Disagree 1 2 3 4 5 Agree
I could have learned the material more effectively using a book.
Disagree 1 2 3 4 5 Agree
I always know how well I am doing.
Disagree 1 2 3 4 5 Agree

PART II: Comparing both systems:

Which system (choose 1) was most:

Helpful to use	A	B
Efficient to use	A	B
Enjoyable to use	A	B

Please add any comments you have about either system:

d) Methods that use physiological monitoring

It has 2 methods

Eye tracking
Physiological measurement

Eye tracking

- Head or desk mounted equipment tracks the position of the eye
- Eye movement reflects the amount of cognitive processing a display requires
- Measurements include
  - Fixations: eye maintains stable position. Number and duration indicate level of difficulty with display
  - Scan paths: moving straight to a target with a short fixation at the target is optimal

Eye-tracking equipment.

physiological measurements

- Emotional response linked to physical changes
- These may help determine a user's reaction to an interface
- Measurements include:
  - heart activity, including blood pressure, volume and pulse.
  - activity of sweat glands: Galvanic Skin Response (GSR)
  - electrical activity in muscle: electromyogram (EMG)
  - electrical activity in brain: electroencephalogram (EEG)
- some difficulty in interpreting these physiological responses - more research needed

Data Lab Psychophysiology equipment showing some of the sensors

CHOOSING AN EVALUATION METHOD - Factors distinguishing evaluation techniques

- the stage in the cycle at which the evaluation is carried out
- the style of evaluation
- the level of subjectivity or objectivity of the technique
- the type of measures provided
- the information provided
- the immediacy of the response
- the level of interference implied
- the resources required.

2.19 Universal Design Principles:

- Universal design as 'the process of designing products so that they can be used by as many people as possible in as many situations as possible'. These principles give us a framework in which to develop universal designs.
- Principle one is equitable use: the design is useful to people with a range of abilities and appealing to all. No user is excluded or stigmatized.

- Principle two is flexibility in use: the design allows for a range of ability and preference, through choice of methods of use and adaptively to the user's pace, precision and custom.
- Principle three is that the system be simple and intuitive to use, regardless of the knowledge, experience, language or level of concentration of the user.
- Principle four is perceptible information: the design should provide effective communication of information regardless of the environmental conditions or the user's abilities. Redundancy of presentation is important: information should be represented in different forms or modes (e.g. graphic, verbal, text, touch).
    - Essential information should be emphasized and differentiated clearly from the peripheral content. Presentation should support the range of devices and techniques used to access information by people with different sensory abilities.
- Principle five is tolerance for error: minimizing the impact and damage caused by mistakes or unintended behavior. Potentially dangerous situations should be removed or made hard to reach. Potential hazards should be shielded by warnings. Systems should fail safe from the user's perspective and users should be supported in tasks that require concentration.
- Principle six is low physical effort: systems should be designed to be comfortable to use, minimizing physical effort and fatigue. The physical design of the system should allow the user to maintain a natural posture with reasonable operating effort. Repetitive or sustained actions should be avoided.
- Principle seven requires size and space for approach and use: the placement of the system should be such that it can be reached and used by any user regardless of body size, posture or mobility. Important elements should be on the line of sight for both seated and standing users. All physical components should be comfortably reachable by seated or standing users.

2.19.1 MULTI-MODAL INTERACTION (EXTRA)

The principle of universal design relies on multi-modal interaction.

Multi-Sensory Systems

- More than one sensory channel in interaction
    - e.g. sounds, text, hypertext, animation, video, gestures, vision
- Used in a range of applications:
    - particularly good for users with special needs, and virtual reality
- It increases the *bandwidth of the interaction* between the human and the computer
- It makes human–computer interaction more like the interaction between humans and their everyday environment

Usable Senses

The 5 senses (sight, sound, touch, taste and smell) are used by us every day

- each is important on its own
- together, they provide a fuller interaction with the natural world

Computers rarely offer such a rich interaction

Can we use all the available senses?

- ideally, yes

– practically – no

We can use • sight • sound • touch (sometimes)
We cannot (yet) use • taste • smell

Multi-modal vs. Multi-media

- Multi-modal systems
  - use more than one sense (or mode) of interaction

e.g. visual and aural senses: a text processor may speak the words as well as echoing them to the screen

- Multi-media systems
  - use a number of different media to communicate information

e.g. a computer-based teaching system:may use video, animation, text and still images: different media all using the visual mode of interaction; may also use sounds, both speech and non-speech.

Speech

Human beings have a great and natural mastery of speech

- makes it difficult to appreciate the complexities but
- it's an easy medium for communication

Structure of Speech

- Phonemes
  - English contains 40 phonemes- 24 consonants and 16 vowel sounds
  - basic atomic units
  - sound slightly different depending on the context they are in
- Allophones
  - all the sounds in the language
  - between 120 and 130 of them
- Morphemes
  - either parts of words or whole words
  - smallest unit of language that has meaning.

The Phonetic Typewriter

- Developed for Finnish (a phonetic language, written as it is said)
- Trained on one speaker, will generalise to others.
- A neural network is trained to cluster together similar sounds, which are then labelled with the corresponding character.
- When recognising speech, the sounds uttered are allocated to the closest corresponding output, and the character for that output is printed.
  - requires large dictionary of minor variations to correct general mechanism
  - noticeably poorer performance on speakers it has not been trained on

Speech Synthesis

The process of generating spoken language by machine on the basis of written input . Useful for users who are blind or partially sighted
Successful in certain constrained applications when the user:

- is particularly motivated to overcome problems
- has few alternatives

Examples:

- screen readers

    - read the textual display to the user
      utilised by visually impaired people
- warning signals
    - spoken information sometimes presented to pilots whose visual and haptic skills are already fully occupied

Non-Speech Sounds

- Often used to provide transitory information, such as indications of network or system changes, or of errors
- Used to provide status information on background processes
- Commonly used for warnings and alarms
- Evidence to show they are useful
    - fewer typing mistakes with key clicks
    - video games harder without sound
- Language/culture independent, unlike speech

Touch

- Use of touch in the interface is known as *haptic interaction*
    - Cutaneous perception
        - tactile sensation(the **sensation** produced by pressure receptors in the skin); vibrations on the skin
    - kinesthetics
        - movement and position; force feedback
- Information on shape, texture, resistance, temperature, comparative spatial factors
- example technologies
    - electronic braille displays
    - force feedback devices Eg.PHANTOM
        - resistance, texture

Handwriting recognition

- Handwriting is another communication mechanism which we are used to in day-to-day life
- Interpret handwritten input and handwriting appears to offer both textual and graphical input
- Technology
    - Handwriting consists of complex strokes and spaces
    - Captured by digitizing tablet
        - strokes transformed to sequence of dots
    - large tablets available
        - suitable for digitizing maps and technical drawings
    - smaller devices, some incorporating thin
      screens to display the information
        - PDAs such as Palm Pilot
        - tablet PCs
- Problems
    - personal differences in letter formation
    - co-articulation effects
- Breakthroughs:
    - stroke not just bitmap
    - special 'alphabet' – Graffeti on PalmOS
- Current state:
    - usable – even without training
    - but many prefer keyboards!

Gesture

- Able to control the computer with certain movements of the hand
- Applications
  - gestural input - e.g. “put that there”
  - sign language
- Technology
  - data glove
  - position sensing devices
- Benefits
  - natural form of interaction - pointing
  - enhance communication between signing and non-signing users
- Problems
  - user dependent, variable and issues of coarticulation

Users with disabilities

- visual impairment
  - screen readers, SonicFinder
- hearing impairment
  - text communication, gesture, captions
- physical impairment
  - speech I/O, gesture, predictive systems (e.g. Reactive keyboard)
- speech impairment
  - speech synthesis, text communication
- age groups
  - older people e.g. disability aids, memory aids, communication tools to prevent social isolation
  - children e.g. appropriate input/output devices, involvement in design process
- cultural differences
  - influence of nationality, generation, gender, race, sexuality, class, religion, political persuasion etc. on interpretation of interface features
  - e.g. interpretation and acceptability of language, cultural symbols, gesture and colour