(Human-Computer Interaction)

IS252

M.M IMAN M. HASSAN

Chapter 1

Interaction Design

1.1 Introduction

In this chapter, we begin by examining what interaction design is. We look at the difference between good and poor design, highlighting how products can differ radically in their usability. We then describe what and who is involved in interaction design. In the last part of the chapter we outline core aspects of usability and how these are used to assess interactive products. An assignment is presented at the end of the chapter in which you have the opportunity to put into practice what you have read, by evaluating an interactive product using various usability criteria. The main aims of the chapter are to:

1. Explain the difference between good and poor interaction design.
2. Describe what interaction design is and how it relates to human-computer interaction and other fields.
3. Explain what usability is.
4. Describe what is involved in the process of interaction design.
5. Outline the different forms of guidance used in interaction design.
6. Enable you to evaluate an interactive product and explain what is good and bad about it in terms of the goals and principles of interaction design.

1.2 Good and poor design

A central concern of interaction design is to develop interactive products that are sable. By this is generally meant easy to learn, effective to use, and provide an enjoyable user experience. A good place to start thinking about how to design usable interactive products is to compare examples of well and poorly designed ones.

Through identifying the specific weaknesses and strengths of different interactive systems, we can begin to understand what it means for something to be usable or not. Here, we begin with an example of a poorly designed system -voice mail- that is used in many organizations (businesses, hotels, and universities). We then compare this with an answering machine that exemplifies good design.

1.2.1 What to design

Designing usable interactive products thus requires considering who is going to be using them and where they are going to be used. Another key concern is under- standing the kind of activities people are doing when interacting with the products. The appropriateness of different kinds of interfaces and arrangements of input and output devices depends on what kinds of activities need to be supported. A key question for interaction design is: how do you optimize the users' inter-actions with a system, environment or product, so that they match the users' activities that are being supported and extended? One could use intuition and hope for the best.

Alternatively, one can be more principled in deciding which choices to make by basing them on an understanding of the users. This involves:

1. taking into account what people are good and bad at considering what might help people with the way they currently do things
2. thinking through what might provide quality user experiences
3. listening to what people want and getting them involved in the design using "tried and tested" user-based techniques during the design process

1.3 What is interaction design
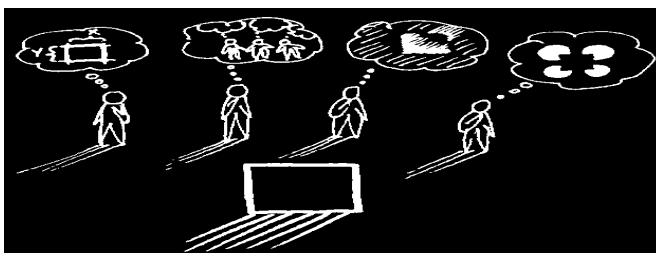
By interaction design, we mean

"Designing interactive products to support people in their everyday and working lives".

1.3.1 The makeup of interaction design

One of the biggest challenges at that time was to develop computers that could be accessible and usable by other people, besides engineers, to support tasks involving human cognition (e.g., doing sums, writing documents, managing accounts, drawing plans). To make this possible, computer scientists and psychologists became involved in designing user interfaces. Computer scientists and software engineers developed high-level programming languages (e.g., BASIC, Prolog), system architectures, software design methods, and command-based languages to help in such tasks, while psychologists provided information about human capabilities (e.g., memory, decision making).

1.3.2 Working together as a multidisciplinary team

Bringing together so many people with different backgrounds and training has meant many more ideas being generated, new methods being developed, and more creative and original designs being produced. However, the down side is the costs involved. The more people there are with different backgrounds in a design team, the more difficult it can be to communicate and progress forward the designs being generated. Why? People with different backgrounds have different perspectives and ways of seeing and talking about the world (see Figure 1.4).

Figure 1.4 Four different team members looking at the same square, but each Seeing it quite differently.

1.3.3 Interaction design in business

Interaction design is now a big business. In particular, website consultants, startup companies, and mobile computing industries have all realized its pivotal role in successful interactive products. To get noticed in the highly competitive field of web products requires standing out. Being able to say that your product is easy and effective to use is seen as central to this.

BOX 1.2 What's In a Name? From Interface Designers to Information Architects

Ten years ago, when a company wanted to develop an interface for an interactive product it advertised for interface designers. Such professionals were primarily involved in the design and evaluation of widgets for desktop applications. Now that the potential range of interactive products has greatly diversified, coupled with the growing realization of the importance of getting the interface right, a number of other job descriptions have begun to emerge. These include:

- interactive/interaction designers (people involved in the design of all the interactive aspects of a product, not just the graphic design of an interface)
- usability engineers (people who focus on evaluating products, using usability methods and principles)
- web designers (people who develop and create the visual design of websites, such as layouts)
- information architects (people who come up with ideas of how to plan and structure interactive products, especially websites)
- user-experience designers (people who do all the above but who may also carry out field studies to inform the design of products)

1.4 What is involved in the process of interaction design?

Essentially, the process of interaction design involves four basic activities:

1. Identifying needs and establishing requirements.

2. Developing alternative designs that meet those requirements.

3. Building interactive versions of the designs so that they can be communicated and assessed.

4. Evaluating what is being built throughout the process.

These activities are intended to inform one another and to be repeated. For example, measuring the usability of what has been built in terms of whether it is easy to use provides feedback that certain changes must be made or that certain requirements have not yet been met.

In addition to the four basic activities of design, there are three key characteristics of the interaction design process:

1. Users should be involved through the development of the project.

2. Specific usability and user experience goals should be identified, clearly documented, and agreed upon at the beginning of the project.

3. Iteration through the four activities is inevitable.

1.5 The goals of interaction design

Part of the process of understanding users' needs, with respect to designing an interactive system to support them, is to be clear about your primary objective. Is it to design a very efficient system that will allow users to be highly productive in their work, or is it to design a system that will be challenging and motivating so that it supports effective learning, or is it something else? We call these top level concerns usability goals and user experience goals. The two differ in terms of how they are operationalized, how they can be met and through what means. Usability goals are concerned with meeting specific usability criteria (e.g., efficiency) and user experience goals are largely concerned with explicating the quality of the user experience (e.g., to be aesthetically pleasing).

1.5.1 Usability goals

To recap, usability is generally regarded as ensuring that interactive products are easy to learn, effective to use, and enjoyable from the user's perspective. It involves optimizing the interactions people have with interactive products to enable them to carry out their activities at work, school, and in their everyday life. More specifically, usability is broken down into the following goals:

- effective to use (effectiveness)
- efficient to use (efficiency)
- safe to use (safety)
- have good utility (utility)
- easy to learn (learnability)
- easy to remember how to use (memorability) For each goal, we describe it in more detail

Effectiveness is a very general goal and refers to how good a system is at doing what it is supposed to do.

Efficiency refers to the way a system supports users in carrying out their tasks.

Safety involves protecting the user from dangerous conditions and undesirable situations. In relation to the first ergonomic aspect, it refers to the external conditions where people work

Utility refers to the extent to which the system provides the right kind of functionality so that users can do what they need or want to do.

Learnability refers to how easy a system is to learn to use. It is well known that people don't like spending a long time learning how to use a system. They want to get started straight away and

become competent at carrying out tasks without too much effort. This is especially so for interactive products intended for everyday use (e.g., interactive TV, email) and those used only infrequently (e.g., video conferencing).

Memorability refers to how easy a system is to remember how to use, once learned. This is especially important for interactive systems that are used infrequently. If users haven't used a system or an operation for a few months or longer, they should be able to remember or at least rapidly be reminded how to use it.

1.5.2 User experience goals

The realization that new technologies are offering increasing opportunities for sup- porting people in their everyday lives has led researchers and practitioners to con-sider further goals. The emergence of technologies (e.g., virtual reality, the web, mobile computing) in a diversity of application areas (e.g., entertainment, education, home, public areas) has brought about a much wider set of concerns. As well as focusing primarily on improving efficiency and productivity at work, interaction design is increasingly concerning itself with creating systems that are:

- satisfying
- enjoyable
- fun
- entertaining
- helpful
- motivating
- aesthetically pleasing
- supportive of creativity
- rewarding
- emotionally fulfilling
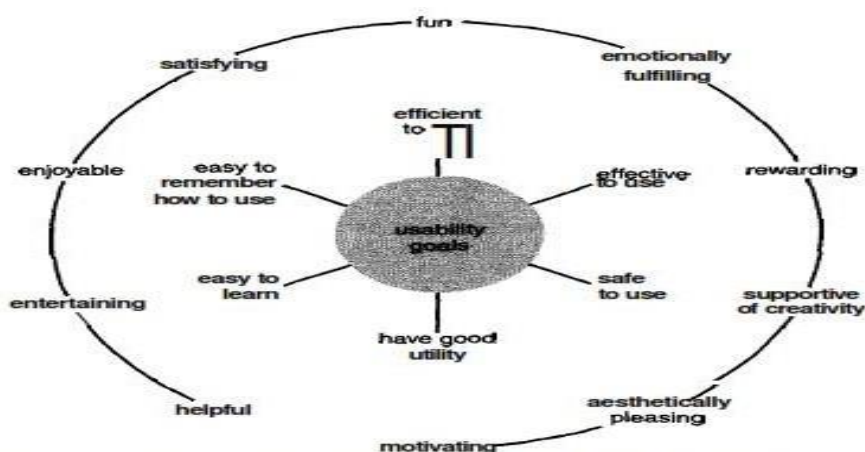
Figure 1.7 Usability and user experience goals. Usability goals are central to interaction de-sign and are operationalized through specific criteria. User experience goals are shown in the outer circle and are less clearly defined.

Chapter 2

Understanding and conceptualizing in interaction

2.1 Introduction

Imagine you have been asked to design an application to let people organize, store, and retrieve their email in a fast, efficient and enjoyable way. What would you do? How would you start? Would you begin by sketching out how the interface might look, work out how the system architecture will be structured, or even just start coding Alternatively, would you start by asking users about their current experiences of saving email, look at existing email tools and, based on this, begin thinking about why, what, and how you were going to design the application?

Interaction designers would begin to doing it. It is important to realize that having a clear understanding of what, why, and how you are going to design something, before writing any code, can save enormous amounts of time and effort later on in the design process. Ill-thought-out ideas, incompatible and unusable designs can be ironed out while it is relatively easy and painless to do. Once ideas are committed to code (which typically takes considerable effort, time, and money), they become much harder to throw away and much more painful.

The main aims of this chapter are to:
1. Explain what is meant by the problem space.
2. Explain how to conceptualize interaction.
3. Describe what a conceptual model is and explain the different kinds.
4. Discuss the pros and cons of using interface metaphors as conceptual models.

2.2   Understanding the problem space

In the process of creating an interactive product, it can be tempting to begin at the "nuts and bolts" level of the design. By this, we mean working out how to design the physical interface and what interaction styles to use (e.g., whether to use menus, forms, speech, icons, or commands).

A problems with trying to solve a design problem beginning at this level is that.

For example, consider the problem of providing drivers with better navigation and traffic information. How might you achieve this? One could tackle the problem by thinking straight away about a good technology or kind of interface to use. since it can be useful for integrating additional information with an ongoing activity, it could be effective for displaying information to drivers who need to find out where they are going and what to do at certain points during their journey. In particular, images of places and directions to follow could be projected inside the car, on the dashboard or rear-view mirror. However, there is a major problem with this proposal: it is likely to be very unsafe. It could easily distract drivers, luring them to switch their attention from the road to where the images were being projected.

A problem in starting to solve a design problem at the physical level, therefore, is that usability goals can be easily overlooked. it is better to make these kinds of design decisions after understanding the nature of the problem space. By this, we mean conceptualizing what you want to create and articulating why you want to do so. This requires thinking through how your design will support people in their everyday or work activities. In particular, you need to ask yourself whether the interactive product you have in mind will achieve what you hope it will. If so, how? In the above example, this involves finding out what is problematic with existing forms of navigating while driving (e.g., trying to read maps while moving the steering wheel) and how to ensure that drivers can continue to drive safely without being distracted.

A framework for explicating assumptions

Reasoning through your assumptions about why something might be a good idea enables you to see the strengths and weaknesses of your proposed design. In so doing, it enables you to be in a better position to commence the design process. We have shown you how to begin this, through operationalizing relevant usability goals. In addition, the following questions provide a useful framework with which to begin thinking through the problem space:

- Are there problems with an existing product? If so, what are they? Why do you think there are problems?
- Why do you think your proposed ideas might be useful? How do you envision people integrating your proposed design with how they currently do things in their everyday or working lives?
- How will your proposed design support people in their activities? In what way does it address an identified problem or extend current ways of doing things? Will it really help?

2.3   Conceptual models

"The most important thing to design is the user's conceptual model. Everything else should be subordinated to making that model clear, obvious, and substantial. That is almost exactly the opposite of how most software is designed."

By a conceptual model is meant:

a description of the proposed system in terms of a set of integrated ideas and concepts about what it should do, behave and look like, that will be understandable by the users in the manner intended.

Once a set of possible ways of interacting with an interactive system has been identified, the design of the conceptual model then needs to be through in terms of actual concrete solutions. This

entails working out the behavior of the interface, the particular interaction styles that will be used, and the "look and feel" of the interface. At this stage of "fleshing out," it is always a good idea to explore a number of possible designs and to assess the merits and problems of each one.

Another way of designing an appropriate conceptual model is to select an interface metaphor. This can provide a basic structure for the conceptual model that is couched in knowledge users are familiar with. Examples of well-known interface metaphors are the desktop and search engines.

Here, we describe the different kinds of conceptual models, interface metaphors, and interaction paradigms to give you a good understanding of the various types prior to thinking about how to design them.

There are a number of different kinds of conceptual models. These can be broken down into two main categories: those based on activities and those based on objects.

2.3.1 Conceptual models based on activities

The most common types of activities that users are likely to be engaged in when interacting with systems are:

1. instructing
2. conversing
3. manipulating and navigating
4. exploring and browsing

A first thing to note is that the various kinds of activity can be carried out together. For example, it is possible for someone to give instructions while conversing or navigate an environment while browsing. However, each has different properties and suggests different ways of being developed at the interface.

The first one is based on the idea of letting the user issue instructions to the system when performing tasks. This can be done in various interaction styles: typing in commands, selecting options from menus in a windows environment or on a touch screen, speaking aloud commands, pressing buttons, or using a combination of function keys.

1. Instructing: This kind of conceptual model describes how users carry out their tasks through instructing the system what to do. Examples include giving instructions to a system to perform operations like tell the time, print a file, and remind the user of an appointment. A diverse range of devices designed based on this model, include hi-fi systems, alarm clocks, computers.

The second one is based on the user conversing with the system as though talking to someone else. Users speak to the system or type in questions to which the system replies via text or speech output.

2. Conversing: This conceptual model is based on the idea of a person conversing with a system,

where the system acts as a dialog partner. In particular, the system is designed to respond in a way another human being might when having a conversation with someone else. It differs from the previous category of instructing in being intended to reflect a more two-way communication process, where the system acts more like a partner than a machine that simply obeys orders. This kind of conceptual model has been found to be most useful for applications in which the user needs to find out specific kinds of information or wants to discuss issues. Examples include advisory systems, help facilities, and search engines. The proposed tourist application described earlier would fit into this category.

The third type is based on allowing users to manipulate and navigate their way through an environment of virtual objects. It assumes that the virtual environment shares some of the properties of the physical world, allowing users to use their knowledge of how physical objects behave when interacting with virtual objects.

3. Manipulating and navigating: This conceptual model describes the activity of manipulating objects and navigating through virtual spaces by exploiting users' knowledge of how they do this in the physical world. For example, virtual objects can be manipulated by moving, selecting, opening, closing, and zooming in and out of them. Extensions to these actions can also be included, such as manipulating objects or navigating through virtual spaces, in ways not possible in the real world. For example, some virtual worlds have been designed to allow users to teleport from place to place or to transform one object into another.
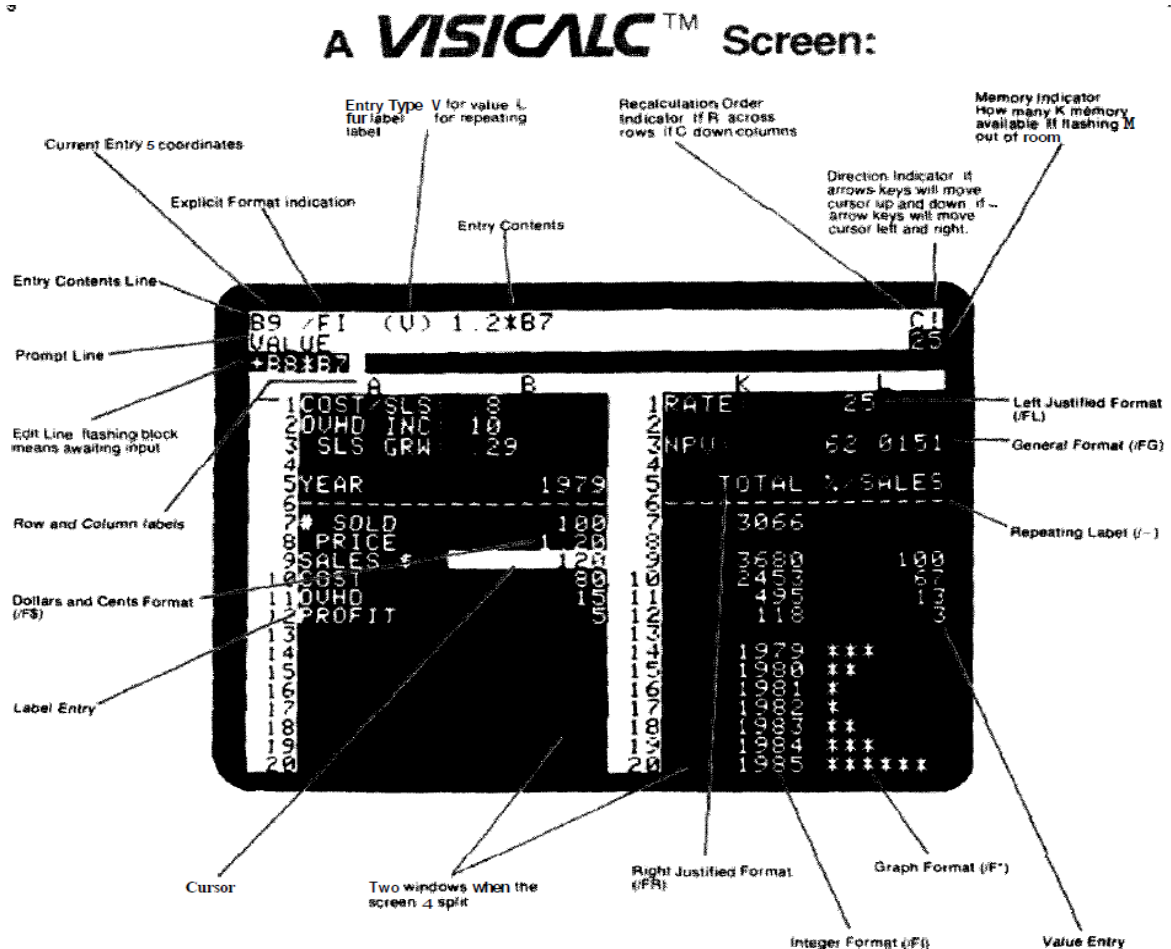
The fourth kind is based on the system providing information that is structured in such a way as to allow users to find out or learn things, without having to formulate specific questions to the system.

4. Exploring and browsing: This conceptual model is based on the idea of allowing people to explore and browse information, exploiting their knowledge of how they do this with existing media (e.g., books, magazines, TV, radio, libraries, pamphlets, brochures). When people go to a tourist office, a bookstore, or a dentist's surgery, often they scan and flick through parts of the information displayed, hoping to find something interesting to read. CD-ROMs, web pages, portals and e-commerce sites are applications based on this kind of conceptual model. Much thought needs to go into structuring the information in ways that will support effective navigation, allowing people to search, browse, and find different kinds of information.

2.3.2 Conceptual models based on objects

The second category of conceptual models is based on an object or artifact, such as a tool, a book, or a vehicle. These tend to be more specific than conceptual models based on activities, focusing on the way a particular object is used in a particular context. They are often based on an

analogy with something in the physical world. An example of a highly successful conceptual model based on an object is the spreadsheet (Winograd, 1996).

A **VISICALC**™ Screen:

Current Entry s coordinates
Entry Type V for value L for label
Explicit Format indication
Entry Contents
Recalculation Order Indicator if R across rows if C down columns
Memory Indicator How many K memory available if flashing M out of room
Direction Indicator if arrows keys will move cursor up and down if ~ arrow keys will move cursor left and right.
Entry Contents Line
Prompt Line
Edit Line flashing block means awaiting input
Row and Column labels
Dollars and Cents Format (/F$)
Label Entry
Left Justified Format (/FL)
General Format (/FG)
Repeating Label (/~)
Cursor
Two windows when the screen 4 split
Right Justified Format (/FR)
Graph Format (/F*)
Integer Format (/FI)
Value Entry

Figure 2.7 Reference card showing annotated screen dump for VisiCalc

The object this is based on is the ledger sheet. The first spreadsheet was designed by Dan Bricklin, and called VisiCalc. It enabled people to carry out a range of tasks that previously could only be done very laboriously and with much difficulty using other software packages, a calculator, or by hand (see Figure 2.7). The main reasons why the spreadsheet has become so successful are first, that Bricklin understood what kind of tool would be useful to people in the financial world (like accountants) and second, he knew how to design it so that it could be used in the way that these people would find useful. Thus, at the outset, he understood

(i)   the kinds of activities involved in the financial side of business, and

(ii)   the problems people were having with existing tools when trying to achieve these activities.

2.4 Interface metaphors

Another way of describing conceptual models is in terms of interface metaphors. By this is meant a conceptual model that has been developed to be similar in some way to aspects of a

physical entity (or entities) but that also has its own behaviors and properties. Such models can be based on an activity or an object or both. As well as being categorized as conceptual models based on objects, the desktop and the spreadsheet are also examples of interface metaphors.

Another example of an interface metaphor is a "search engine." The tool has been designed to invite comparison with a physical object-a mechanical engine with several parts working-together with an everyday action-searching by looking through numerous files in many different places to extract relevant information.

Interface metaphors are based on conceptual models that combine familiar knowledge with new concepts. As mentioned in Box 2.2, the Star was based on a conceptual model of the familiar knowledge of an office. Paper, folders, filing cabinets, and mailboxes were represented as icons on the screen and were designed to possess some of the properties of their physical counterparts. Dragging a document icon across the desktop screen was seen as equivalent to picking up a piece of paper in the physical world and moving it (but of course is a very different action). Similarly, dragging an electronic document onto an electronic folder was seen as being analogous to placing a physical document into a physical cabinet. In addition, new concepts that were incorporated as part of the desktop metaphor were operations that couldn't be performed in the physical world. For example, electronic files could be placed onto an icon of a printer on the desktop, resulting in the computer printing them out.

Benefits of interface metaphors

Interface metaphors have proven to be highly successful, providing users with a familiar orienting device and helping them understand and learn how to use a system. People find it easier to learn and talk about what they are doing at the computer interface in terms familiar to them-whether they are computer-phobic or highly experienced programmers.

Interaction styles

Interaction can be seen as a dialog between the computer and the user. The choice of interface style can have a profound effect on the nature of this dialog. Here we introduce the most common interface styles and note the different effects these have on the interaction.

There are a number of common interface styles including:
1. Command line interface
2. Menus
3. Natural language
4. Question/answer and query dialog

5. Form-fills and spreadsheets
6. WIMP
7. Point and click
8. Three-dimensional interfaces.

As the WIMP interface is the most common and complex, we will discuss each of its elements in greater detail in Section 3.6.

1) **Command line interface:** The command line interface (Figure 3.7) was the first interactive dialog style to be commonly used and, in spite of the availability of menu-driven interfaces, it is still widely used. It provides a means of expressing instructions to the computer directly, using function keys, single characters, abbreviations or whole-word commands.

```
sable.soc.staffs.ac.uk> javac HelloWorldApp
javac: invalid argument: HelloWorldApp
use: javac [-g][-O][-classpath path][-d dir] file.java…
sable.soc.staffs.ac.uk> javac HelloWorldApp.java
sable.soc.staffs.ac.uk> java HelloWorldApp
Hello world!!
sable.soc.staffs.ac.uk>
```

Figure 3.7 Command line interface

2) **Menus:** In a menu-driven interface, the set of options available to the user is displayed on the screen, and selected using the mouse, or numeric or alphabetic keys. Since the options are visible they are less demanding of the user, relying on recognition rather than recall.

```
PAYMENT DETAILS                P3-7

please select payment method:
   1. cash
   2. check
   3. credit card
   4. invoice

   9. abort transaction
```

Figure 3.8 Menu-driven interface

3) **Natural language:** Perhaps the most attractive means of communicating with computers, at least at first glance, is by natural language. Users, unable to remember a command or lost in a hierarchy of menus, may long for the computer that is able to understand instructions

expressed in everyday words! Natural language understanding, both of speech and written input. Unfortunately, however, the ambiguity of natural language makes it very difficult for a machine to understand. Language is ambiguous at a number of levels. First, the syntax, or structure, of a phrase may not be clear. If we are given the sentence

*The boy hit the dog with the stick*

We cannot be sure whether the boy is using the stick to hit the dog or whether the dog is holding the stick when it is hit. Even if a sentence's structure is clear, we may find ambiguity in the meaning of the words used.

Given these problems, it seems unlikely that a general natural language interface will be available for some time. The system can be provided with sufficient information to disambiguate terms. It is important in interfaces which use natural language in this restricted form that the user is aware of the limitations of the system and does not expect too much understanding.

4) **Question/answer and query dialog:** dialog is a simple mechanism for providing input to an application in a specific domain. The user is asked a series of questions (mainly with yes/no responses, multiple choice, or codes) and so is led through the interaction step by step. An example of this would be web questionnaires. These interfaces are easy to learn and use, but are limited in functionality and power. As such, they are appropriate for restricted domains (particularly information systems) and for novice or casual users. Query languages, on the other hand, are used to construct queries to retrieve information from a database. They use natural language-style phrases, but in fact require specific syntax, as well as knowledge of the database structure.

5) **Form-fills interfaces:** are used primarily for data entry but can also be useful in data retrieval applications. The user is presented with a display resembling a paper form, with slots to fill in (see Figure 3.9). Often the form display is based upon an actual form with which the user is familiar, which makes the interface easier to use. The user works through the form, filling in appropriate values. The data are then entered into the application in the correct place.
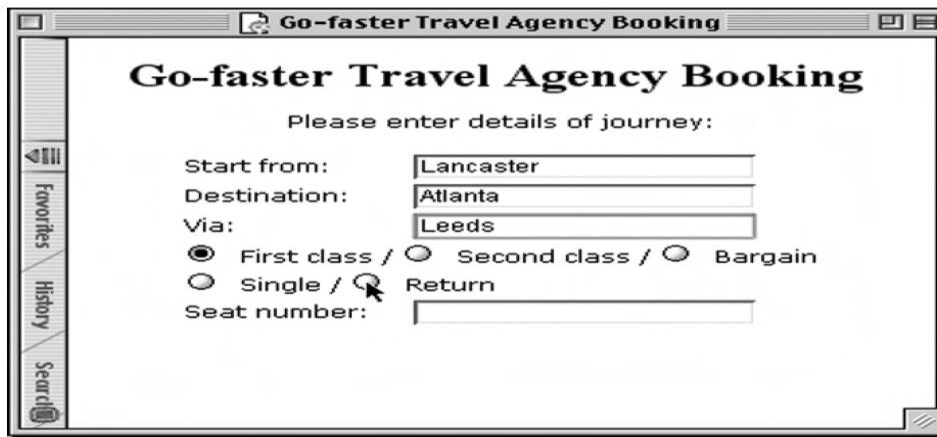
Figure 3.9 A typical form-filling interface. Screen shot frame reprinted by permission from Microsoft Corporation

6) **The WIMP interface :** Currently many common environments for interactive computing are examples of the WIMP interface style, often simply called windowing systems. WIMP stands for windows, icons, menus and pointers (sometimes windows, icons, mice and pull-down menus), and is the default interface style for the majority of interactive computer systems in use today, especially in the PC and desktop workstation arena. Examples of WIMP interfaces include Microsoft Windows for IBM PC compatibles, MacOS for Apple Macintosh compatibles and various X Windows-based systems for UNIX.

7) **Point-and-click interfaces:** In most multimedia systems and in web browsers, virtually all actions take only a single click of the mouse button. You may point at a city on a map and when you click a window opens, showing you tourist information about the city. You may point at a word in some text and when you click you see a definition of the word. You may point at a recognizable iconic button and when you click some action is performed.

8) **Three-dimensional interfaces :** There is an increasing use of three-dimensional effects in user interfaces. The most obvious example is virtual reality, but VR is only part of a range of 3D techniques available to the interface designer.
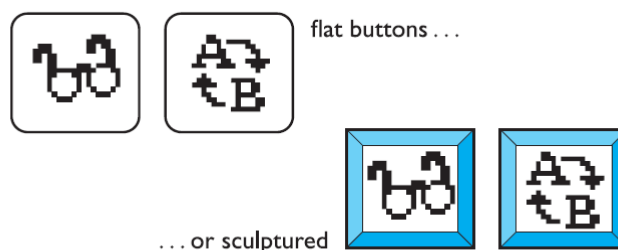
Figure 3.12 Buttons in 3D say 'press me'

Novice users must learn that an oval area with a word or picture in it is a button to be pressed, but a 3D button says 'push me'. Further, more complete 3D environments invite one to move within the virtual environment, rather than watch as a spectator.

Chapter 3

Understanding users

3.1 Introduction

In this chapter we examine some of the core cognitive aspects of interaction design. Specifically, we consider what humans are good and bad at and show how this knowledge can be used to inform the design of technologies that both extend human capabilities and compensate for their weaknesses. We also look at some of the influential cognitively based conceptual frameworks that have been developed for explaining the way humans interact with computers. (Other ways of conceptualizing human behavior that focus on the social and affective aspects of interaction design are presented in the following two chapters.) The main aims of this chapter:

- Explain what cognition is and why it is important for interaction design.
- Describe the main ways cognition has been applied to interaction design.
- Provide a number of examples in which cognitive research has led to the design of more effective interactive products.
- Explain what mental models.
- Give examples of conceptual frameworks that are useful for interaction design.
- Enable you to try to elicit a mental model and be able to understand what it means.

3.2 What is cognition?

"Cognition is what goes on in our heads when we carry out our everyday activities".

It involves cognitive processes, like thinking, remembering, learning, daydreaming, decision making, seeing, reading, writing and talking. As Figure 3.1 indicates, there are many different kinds of cognition. Norman (1993) distinguishes between two general modes: experiential and reflective cognition. The former is a state of mind in which we perceive, act, and react to events around us effectively and effortlessly. It requires reaching a certain level of expertise and engagement. Examples include driving a car, reading a book, having a conversation, and playing a video game. In contrast, reflective cognition involves thinking, comparing, and decision-making. This kind of cognition is what leads to new ideas and creativity. Examples include designing, learning, and writing a book. Norman points out that both modes are essential for everyday life but that each requires different kinds of technological support.

Figure 3.1 What goes on in the mind?

Cognition has also been described in terms of specific kinds of processes. These include:

- attention
- perception and recognition
- memory
- learning
- reading, speaking, and listening
- problem solving, planning, reasoning, decision making

❖ *Attention* is the process of selecting things to concentrate on, at a point in time, from the range of possibilities available. Attention involves our auditory and/or visual senses. An example of auditory attention is waiting in the dentist's waiting room for our name to be called out to know when it is our time to go in. An example of attention involving the visual senses is scanning the football results in a newspaper to attend to information about how our team has done. Attention allows us focus on information that is relevant to what we are doing. The extent to which this process is easy or difficult depends on (i) whether we have clear goals and (ii) whether the information we need is salient in the environment:

(i) *Our goals* If we know exactly what we want to find out, we try to match this with the information that is available.

(ii) *Information presentation* the way information is displayed can also greatly influence how easy or difficult it is to attend to appropriate pieces of information.

❖ *Perception* refers to how information is acquired from the environment, via the different sense organs (e.g., eyes, ears, fingers) and transformed into experiences of objects, events, sounds, and tastes (Roth, 1986). It is a complex process, involving other cognitive processes such as memory, attention, and language. Vision is the most dominant sense for sighted individuals, followed by hearing and touch. With respect to interaction design, it is important to present information in a way that can be readily perceived in the manner intended. For example, there are many ways to design icons. The key is to make them easily distinguishable from one another and to make it simple to recognize what they are intended to represent.

❖ *Memory* involves recalling various kinds of knowledge that allow us to act appropriately. It is very versatile, enabling us to do many things. For example, it allows us to recognize someone's face, remember someone's name, recall when we last met them and know what we said to them last. Simply, without memory we would not be able to function.

❖ *Learning* can be considered in terms of (i) how to use a computer-based application or (ii)

using a computer-based application to understand a given topic. Jack Carroll (1990) and his colleagues have written extensively about how to design interfaces to help learners develop computer-based skills.

A main observation is that people find it very hard to learn by following sets of instructions in a manual. Instead, they much prefer to "learn through doing." GUIs and direct manipulation interfaces are good environments for supporting this kind of learning by supporting exploratory interaction and importantly allowing users to "undo" their actions, i.e., return to previous state if they make a mistake by clicking on the wrong option. Carroll has also suggested that another way of helping learners is by using a "training-wheels approach. This involves restricting the possible functions that can be carried out by a novice to the basics and then extending these as the novice becomes more experienced. The underlying rationale is to make initial learning more tractable, helping the learner focus on simple operations before moving on to more complex ones.

❖ ***Reading, speaking and listening:*** these three forms of language processing have both similar and different properties. One similarity is that the meaning of sentences or phrases is the same regardless of the mode in which it is conveyed. For example, the sentence "Computers are a wonderful invention" essentially has the same meaning whether one reads it, speaks it, or hears it. However, the ease with which people can read, listen, or speak differs depending on the person, task, and context. For example, many people find listening much easier than reading. Specific differences between the three models include:

- Written language is permanent while listening is transient. It is possible to reread information if not understood the first-time round. This is not possible with spoken information that is being broadcast.
- Reading can be quicker than speaking or listening, as written text can be rapidly scanned in ways not possible when listening to serially presented spoken words.
- Listening requires less cognitive effort than reading or speaking. Children, especially, often prefer to listen to narratives provided in multimedia or
- web-based learning material than to read the equivalent text online.
- Written language tends to be grammatical while spoken language is often ungrammatical. For example, people often start a sentence and stop in
- mid-sentence, letting someone else start speaking.
- There are marked differences between people in their ability to use language. Some people prefer reading to listening, while others prefer listening. Likewise, some people prefer speaking to writing and vice versa.
  - Dyslexics have difficulties understanding and recognizing written words,
   making it hard for them to write grammatical sentences and spell correctly.

People who are hard of hearing or hard of seeing are also restricted in the way they can process language

❖ ***Problem-solving, planning, reasoning and decision-making*** are all cognitive processes involving reflective cognition. They include thinking about what to do, what the options are, and what the consequences might be of carrying out a given action. They often involve conscious processes (being aware of what one is thinking about), discussion with others (or oneself), and the use of various kinds of artifacts, (e.g., maps, books, and pen and paper).

Comparing different sources of information is also common practice when seeking information on the web. For example, just as people will phone around for a range of quotes, so too, will they use different search engines to find sites that give the best deal or best information. If people have knowledge of the pros and cons of different search engines, they may also select different ones for different kinds of queries. For example, a student may use a more academically oriented one when looking for information for writing an essay, and a more commercially based one when trying to find out what's happening in town.

3.3 Conceptual frameworks for cognition

In this section we examine three of people's coping strategies in the physical world to the digital world., which each have a different perspective on cognition:

- mental models
- information processing
- external cognition

3.3.1 Mental models

What happens when people are learning and using a system is that they develop knowledge of how to use the system and, to a lesser extent, how the system works. These two kinds of knowledge are often referred to as a user's mental model.

Having developed a mental model of an interactive product, it is assumed that people will use it to make inferences about how to carry out tasks when using the interactive product. Mental models are also used to fathom what to do when something unexpected happens with a system and when encountering unfamiliar systems. The more someone learns about a system and how it functions, the more their mental model develops. For example, TV engineers have a "deep" mental model of how TVs work that allows them to work out how to fix them.

3.3.2 Information processing

Another approach to conceptualizing how the mind works has been to use metaphors and analogies. A number of comparisons have been made, including conceptualizing the mind as

a reservoir, a telephone network, and a digital computer. One prevalent metaphor from cognitive psychology is the idea that the mind is an information processor. Information is thought to enter and exit the mind through a series of ordered processing stages (see Figure 3.11). Within these stages, various processes are assumed to act upon mental representations. Processes include comparing and matching. Mental representations are assumed to comprise images, mental models, rules, and other forms of knowledge.
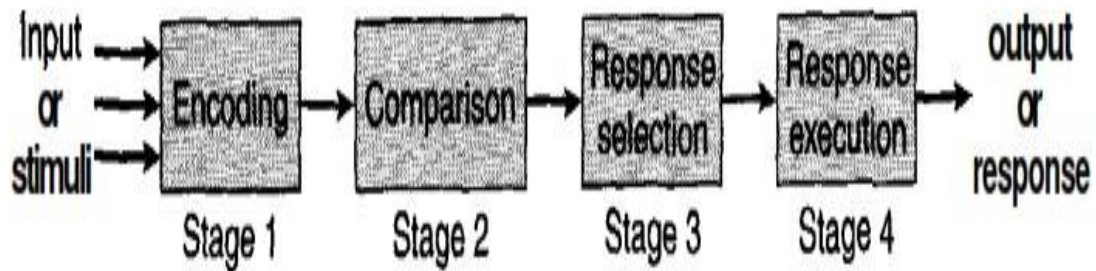
Figure 3.11 Human information processing model.

Several researchers have argued that existing information processing approaches are too impoverished.

The traditional approach to the study of cognition is to look at the pure intellect, isolated from distractions and from artificial aids. Experiments are performed in closed, isolated rooms, with a minimum of distracting lights or sounds, no other people to assist with the task, and no aids to memory or thought. The tasks are arbitrary ones, invented by the researcher. Model builders build simulations and descriptions of these isolated situations.

The theoretical analyses are self -contained little structures, isolated from the world, isolated from any other knowledge or abilities of the person. (Norman, 1990, p. 5)

Instead, there has been an increasing trend to study cognitive activities in the Context in which they occur, analyzing cognition as it happens "in the wild" (Hutchins, 1995). A central goal has been to look at how structures in the environment can both aid human cognition and reduce cognitive load. A number of alternative frameworks have been proposed, including external cognition and distributed cognition.

3.3.3 External cognition

People interact with or create information through using a variety of external representations, e.g., books, multimedia, newspapers, web pages, maps, diagrams, notes, drawings, and so on. Furthermore, an impressive range of tools has been developed

throughout history to aid cognition, including pens, calculators, and computer-based technologies. The combination of external representations and physical tools have greatly extended and supported people's ability to carry out cognitive activities (Norman, 1993). Indeed, they are such an integral part that it is difficult to imagine how we would go about much of our everyday life without them

External cognition is concerned with explaining the cognitive processes involved When we interact with different external representations (Scaife and Rogers, 1996). A main goal is to explicate the cognitive benefits of using different representations for different cognitive activities and the processes involved. The main ones include:

1. externalizing to reduce memory load

2. computational offloading

3. annotating and cognitive tracing

1. Externalizing to reduce memory load

A number of strategies have been developed for transforming knowledge into external representations to reduce memory load. One such strategy is externalizing things we find difficult to remember, such as birthdays, appointments, and addresses. Diaries, personal reminders and calendars are examples of cognitive artifacts that are commonly used for this purpose, acting as external reminders of what we need to do at a given time (e.g., buy a card for a relative's birthday).

Externalizing, therefore, can help reduce people's memory burden by:

- reminding them to do something (e.g., to get something for their mother's birthday)
- reminding them of what to do (e.g., to buy a card)
- reminding them of when to do something (send it by a certain date)

2. Computational offloading

Computational offloading occurs when we use a tool or device in conjunction with an external representation to help us carry out a computation. An example is using pen and paper to solve a math problem.

3. Annotating and cognitive tracing

Another way in which we externalize our cognition is by modifying representations to reflect changes that are taking place that we wish to mark. For example, people often cross

things off in a to-do list to show that they have been completed. They may also reorder objects in the environment, say by creating different piles as the nature of the work to be done changes. These two kinds of modification are called annotating and cognitive tracing:

Annotating involves modifying external representations, such as crossing off or underlining items

3.4 **Informing design: from theory to practice**

Theories, models, and conceptual frameworks provide abstractions for thinking about phenomena. In particular, they enable generalizations to be made about cognition across different situations. For example, the concept of mental models provides a means of explaining why and how people interact with interactive products in the way they do across a range of situations. The information processing model has been used to predict the usability of a range of different interfaces.

Theory in its pure form, however, can be difficult to digest. The arcane terminology and jargon used can be quite off-putting to those not familiar with it. It also requires much time to become familiar with it-something that designers and engineers can't afford when working to meet deadlines.

Researchers have tried to help out by making theory more accessible and practical. This has included translating it into:

- design principles and concepts
- design rules
- analytic methods
- design and evaluation methods

Chapter 4

Process of interaction design

4.1 Introduction

In this chapter, we raise and answer these kinds of questions and discuss the four basic activities and key characteristics of the interaction design process that were introduced in Chapter 1. We also introduce a lifecycle model of interaction design that captures these activities and characteristics.

The main aims of this chapter are to:

1. Consider what 'doing' interaction design involves.
2. Ask and provide answers for some important questions about the interaction design process.
3. Introduce the idea of a lifecycle model to represent a set of activities and how they are related.
4. Describe some lifecycle models from software engineering and HCI and discuss how they relate to the process of interaction design.
5. Present a lifecycle model of interaction design.

4.2 What is interaction design about?

Interaction design involves developing a plan which is informed by the product's intended use, target domain, and relevant practical considerations. Alternative designs need to be generated, captured, and evaluated by users. For the evaluation to be successful, the design must be expressed in a form suitable for users to interact with.

4.2.1 Four basic activities of interaction design

Four basic activities for interaction design were introduced in Chapter 1. These are: identifying needs and establishing requirements, developing alternative designs that meet those requirements, building interactive versions so that they can be communicated and assessed, and evaluating them, i.e., measuring their acceptability. They are fairly generic activities and can be found in other designs disciplines too.

We will be expanding on each of the basic activities of interaction design in the next two chapters. Here we give only a brief introduction to each.

Identifying needs and establishing requirements

In order to design something to support people, we must know who our target users are and what kind of support an interactive product could usefully provide. These needs form the basis of the product's requirements and underpin subsequent design and development. This activity is fundamental to a user centered approach, and is very important in interaction design.

Developing alternative designs

This is the core activity of designing: actually suggesting ideas for meeting the requirements. This activity can be broken up into two sub-activities: conceptual design and physical design. Conceptual design involves producing the conceptual model for the product, and a conceptual model describes what the product should do, behave and look like. Physical design considers the detail of the product including the colors, sounds, and images to use, menu design, and icon design. Alternatives are considered at every point.

Building interactive versions of the designs

Interaction design involves designing interactive products. The most sensible way for users to evaluate such designs, then, is to interact with them. This requires an interactive version of the designs to be built, but that does not mean that a software version is required. There are different techniques for achieving "interaction," not all of which require a working piece of software. For example, paper-based proto- types are very quick and cheap to build and are very effective for identifying problems in the early stages of design, and through role-playing users can get a real sense of what it will be like to interact with the product.

Evaluating designs

Evaluation is the process of determining the usability and acceptability of the product or design that is measured in terms of a variety of criteria including the number of errors users make using it, how appealing it is, how well it matches the requirements, and so on. Interaction design requires a high level of user involvement throughout development, and this enhances the chances

of an acceptable product being delivered. In most design situations you will find a number of activities concerned with quality assurance and testing to make sure that the final product is "fit-for-purpose." Evaluation does not replace these activities, but complements and enhances them.

The activities of developing alternative designs, building interactive versions of the design, and evaluation are intertwined: alternatives are evaluated through the interactive versions of the designs and the results are feedback into further design. This iteration is one of the key characteristics of the interaction design process.

4.2.2 Three key characteristics of the interaction design process

There are three characteristics that we believe should form a key part of the interaction design process. These are: a user focus, specific usability criteria, and iteration. The need to *focus on users* has been emphasized throughout this book, so you will not be surprised to see that it forms a central plank of our view on the interaction design process.

Specific usability and user experience goals should be identified, clearly documented, and agreed upon at the beginning of the project. They help designers to choose between different alternative designs and to check on progress as the product is developed.

Iteration allows designs to be refined based on feedback. As users and designers engage with the domain and start to discuss requirements, needs, hopes and aspirations, then different insights into what is needed, what will help, and what is feasible will emerge.

4.3 Lifecycle models: showing how the activities are related

Understanding what activities are involved in interaction design is the first step to being able to do it, but it is also important to consider how the activities are related to one another so that the full development process can be seen. The term lifecycle model is used to represent a model that captures a set of activities and how they are related. Sophisticated models also incorporate a description of when and how to move from one activity to the next and a description of the deliverables for each activity. The reason such models are popular is that they allow developers, and particularly managers, to get an overall view of the development effort so that progress can be tracked, deliverables specified, resources allocated, targets set, and so on.

4.3.1 A simple lifecycle model for interaction design

We see the activities of interaction design as being related as shown in Figure 6.7. This model incorporates iteration and encourages a user focus. While the outputs from each activity are not specified in the model. Most projects start with identifying needs and

requirements. The project may have arisen because of some evaluation that has been done, but the lifecycle of the new (or modified) product can be thought of as starting at this point. From this activity, some alternative designs are generated in an attempt to meet the needs and requirements that have been identified. Then interactive versions of the designs are developed and evaluated. Based on the feedback from the evaluations, the team may need to return to identifying needs or refining requirements, or it may go straight into redesigning. It may be that more than one alternative design follows this iterative cycle in parallel with others, or it may be that one alternative at a time is considered. Implicit in this cycle is that the final product will emerge in an evolutionary fashion from a rough initial idea through to the finished product.
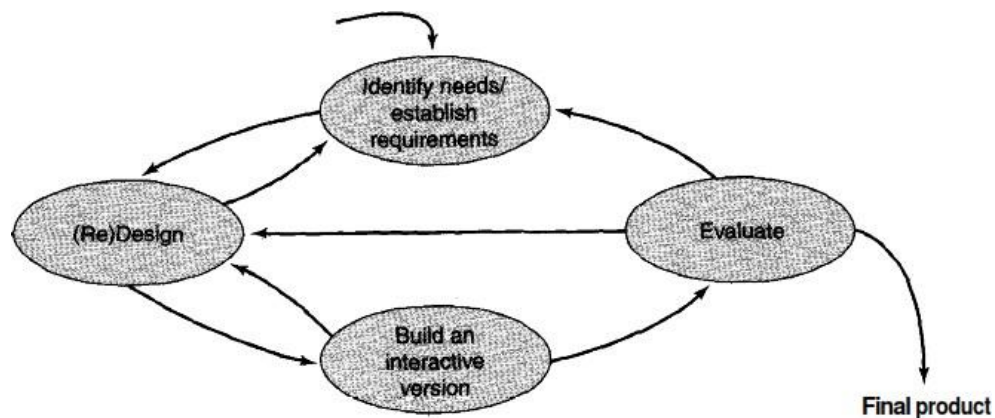
Figure 6.7 A simple interaction design model.

4.3.2 Lifecycle models in software engineering

Software engineering has spawned many lifecycle models, including the waterfall, the spiral, and rapid applications development (RAD).

The waterfall lifecycle model

The waterfall lifecycle was the first model generally known in software engineering and forms the basis of many lifecycles in use today. This is basically a linear model in which each step must be completed before the next step can be started (see Figure 4.2)
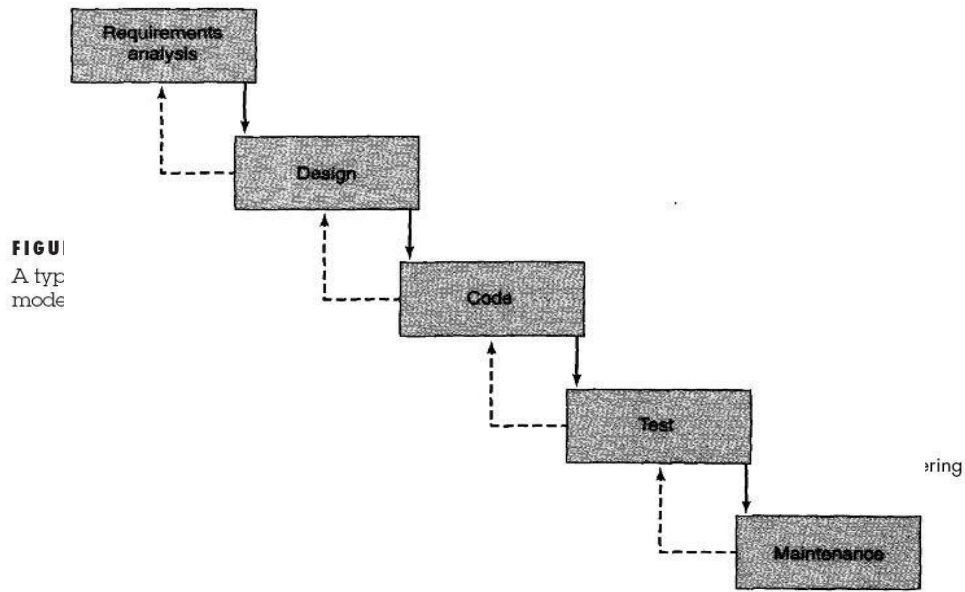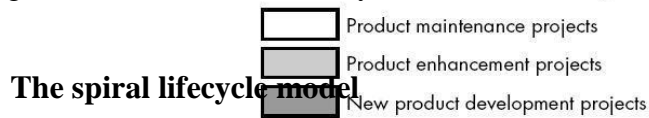
Figure 4.2 The waterfall lifecycle model of software development. The spiral lifecycle model

The spiral lifecycle model

For many years, the waterfall formed the basis of most software developments, but in 1988 Barry Boehm (1988) suggested the spiral model of software development

(see Figure 4.3). Two features of the spiral model are immediately clear from Figure 6.9: risk analysis and prototyping. The spiral model incorporates them in an iterative framework that allows ideas and progress to be repeatedly checked and evaluated. Each iteration around the spiral may be based on a different lifecycle model and may have different activities.

Figure (4.3) the spiral lifecycle model

(RAD) approach attempts to take a user-centered view and to minimize the risk caused by requirements changing during the course of the project. The ideas behind RAD began to

emerge in the early 1990s, also in response to the inappropriate nature of the linear lifecycle models based on the waterfall. Two key features of A RAD project are:

- Time-limited cycles of approximately six months, at the end of which a system or partial system must be delivered. This is called time-boxing. In effect, this breaks down a large project into many smaller projects that can deliver products incrementally, and enhances flexibility in terms of the development techniques used and the maintainability of the final system.

- JAD (Joint Application Development) workshops in which users and developers come together to thrash out the requirements of the system (Wood and Silver, 1995). These are intensive requirements-gathering sessions which difficult issues are faced and decisions are made. Representatives each identified stakeholder group should be involved in each workshop that all the relevant views can be heard.

FIGURE
The RAD model
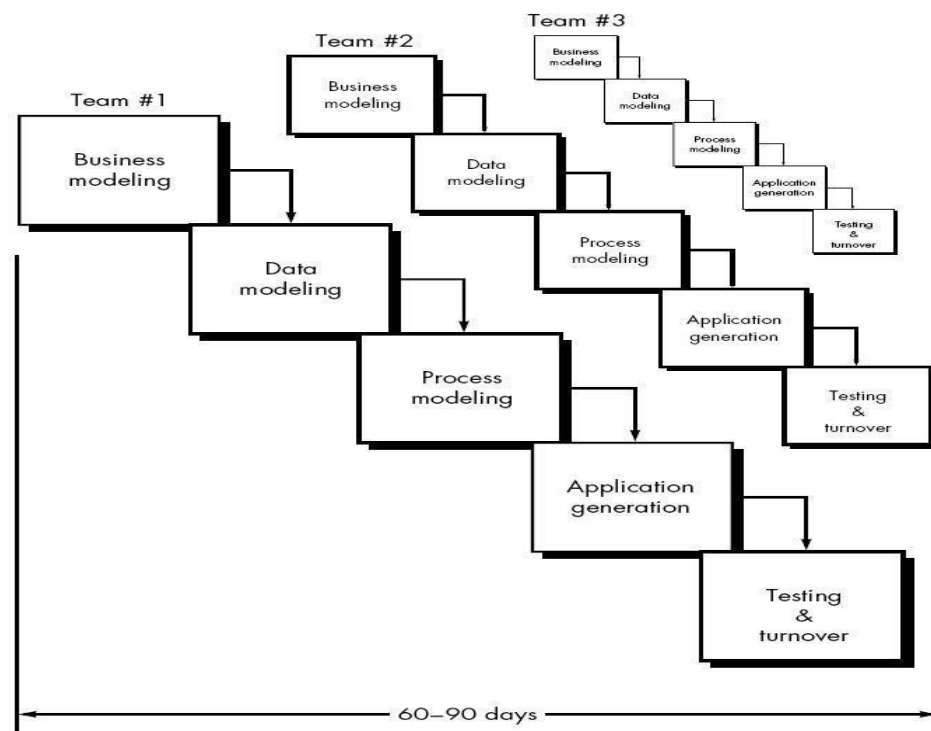
4.3.3 Lifecycle models in HCI

Another of the traditions from which interaction design has emerged is the field of HCI (human -computer interaction). Fewer lifecycle models have arisen from this field than from software engineering and, as you would expect, they have a stronger tradition of user focus. We describe two of these here. The first one, the Star, was derived from empirical

work on understanding how designers tackled HCI design problems. This represents a very flexible process with evaluation at its core. In contrast, the second one, the usability engineering lifecycle, shows a more structured approach and hails from the usability engineering tradition.

The *Star* Lifecycle Model

In 1989, the Star lifecycle model was proposed by Hartson and Hix (1989) (see Figure 4.5). This emerged from some empirical work they did looking at how interface designers went about their work. They identified two different modes of activity: analytic mode and synthetic mode. The former is characterized by such notions as top -down, organizing, judicial, and formal, working from the systems view towards the user's view; the latter is characterized by such notions as bottom-up, free- thinking, creative and ad *hoc,* working from the user's view towards the systems view. Interface designers move from one mode to another when designing a similar behavior has been observed in software designers (Guindon,1990).
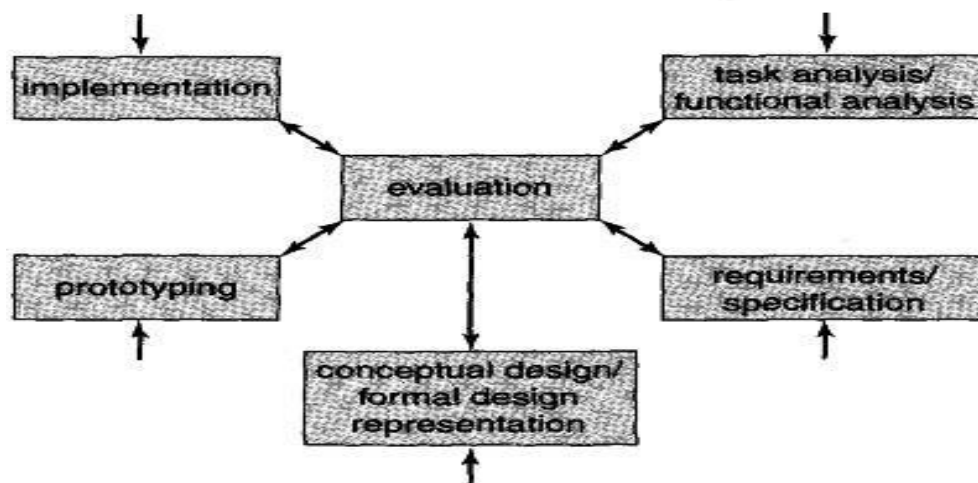

Figure 4.5 the star lifecycle model

The Usability Engineering Lifecycle

The Usability Engineering Lifecycle was proposed by Deborah Mayhew in 1999 (Mayhew, 1999). The lifecycle itself has essentially three tasks: requirements analysis, design, testing, development, and installation, with the middle stage being the largest and involving many subtasks (see Figure 4.6). Note the production of a set of usability goals in the first task. Mayhew suggests that these goals be captured in a style guide that is then used throughout the project to help ensure that the usability goals are adhered to.
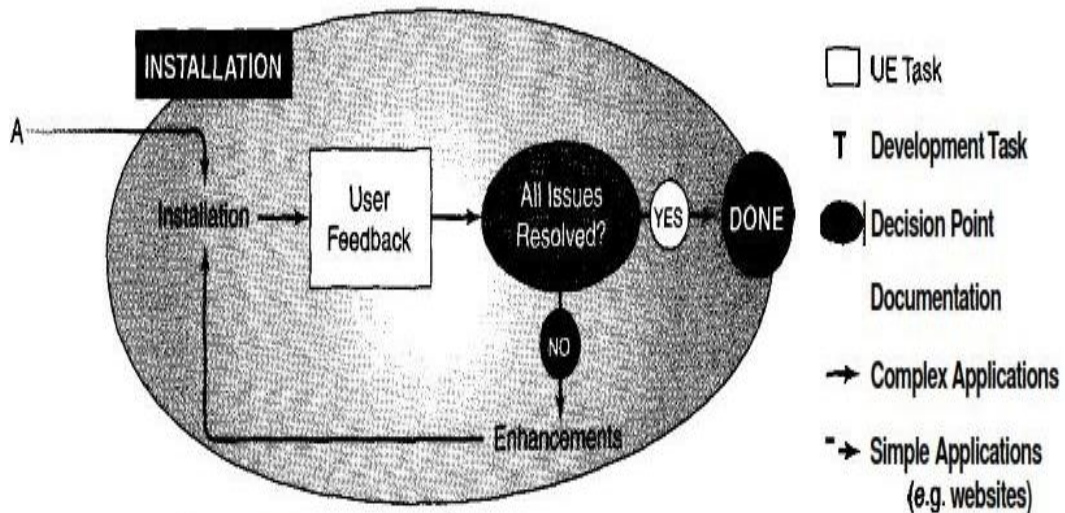
Figure 4.6 the Usability Engineering Lifecycle

Chapter 5:

Identifying needs and establishing requirements

5.1 Introduction

5.2 What, how, and why?

5.3 What are requirements?

5.4 Data gathering

5.5 Data interpretation and analysis

5.1. Introduction

We discussed in Chapter 5, identifying users' needs is not as straightforward as it sounds.

Establishing requirements is also not simply writing a wish list of features. Given the iterative nature of interaction design, isolating requirements activities from design activities and from evaluation activities is a little artificial, since in practice they are all intertwined: some design will take place while requirements are being established, and the design will evolve through a series of evaluation redesign cycles. However, each of these activities can be distinguished by its own emphasis and its own techniques. This chapter provides a more detailed overview of identifying needs and establishing requirements. We introduce different kinds of requirements and explain some useful techniques. The main aims of this chapter are

to: Describe different kinds of requirements. Enable you to identify examples of different kinds of requirements from a simple description. Explain how different data-gathering techniques may be used, and enable you to choose among them for a simple description. Enable you to develop a "scenario," a "use case," and an "essential use case" from a simple description.

5.2. What, how, and why?

5.2.1 What are we trying to achieve in this design activity?

There are two aims.

(i) One aim is to understand as much as possible about the users, their work, and the context of that work, so that the system under development can support them in achieving their goals; this we call "identifying needs."

(ii) Building on this, our second aim is to produce, from the needs identified, a set of stable requirements that form a sound basis to move forward into thinking about design.

This is not necessarily a major document nor a set of rigid prescriptions, but you need to be sure that it will not change radically in the time it takes to do some design and get feedback on the ideas. Because the end goal is to produce this set of requirements, we shall sometimes refer to this as the requirements activity.

5.2.2 How can we achieve this?

At first we give an overview of where we're heading. At the beginning of the requirements activity, we know that we have a lot to find out and to clarify. At the end of the activity we will have a set of stable requirements that can be moved forward into the design activity. In the middle, there are activities concerned with gathering data, interpreting or analyzing1 the data, and capturing the findings in a form that can be expressed as requirements. Broadly speaking, these activities progress in a sequential manner: first gather some data, then interpret it, then extract some requirements from it, but it gets a lot messier than this, and the activities influence one another as the process iterates. One of the reasons for this is that once you start to analyze data, you may find that you need to gather some more data to clarify or confirm some ideas you have. Another reason is that the way in which you document your requirements may affect your analysis,since it will enable you to identify and express some aspects more easily than others.

To overcome this, it is important to use a complementary set of data-gathering techniques and data-interpretation techniques, and to constantly revise and refine the requirements. As we discuss below, there are different kinds of requirements, and each can be emphasized or de-emphasized by the different techniques.

5.3. What are requirements?

A requirement is a statement about an intended product that specifies what it should do or how it should perform. One of the aims of the requirements activity is to make the requirements as specific, unambiguous, and clear as possible.

5.3.1 Different kinds of requirements

In software engineering, two different kinds of requirements have traditionally been identified: ***functional requirements***, which say what the system should do, and ***non-functional requirements***, which say what constraints there are on the system and its development.

For example, a *functional requirement* for a word processor may be that it should support a variety of formatting styles. This requirement might then be decomposed into more specific requirements detailing the kind of formatting required such as formatting by paragraph, by character, and by document, down to a very specific level such as that character formatting must include 20 typefaces, each with bold, italic, and standard options. *A non-functional requirement* for a word processor might be that it must be able to run on a variety of platforms such as PCs, Macs and UNIX machines. Another might be that it must be able to function on a computer with 64 MB RAM. A different kind of non-functional requirement would be that it must be delivered in six months' time. This represents a constraint on the development activity itself rather than on the product being developed.

Interaction design requires us to understand the functionality required and the constraints under which the product must operate or be developed. However, instead of referring to all requirements that are not functional as simply "non-functional" requirements, we prefer to refine this into further following categories:

- Functional requirements capture what the product should do
- Data requirements capture the type, volatility, size amount, persistence, accuracy, and value of the amounts of the required data.
- Environmental requirements or context of use refer to the circumstances in which the interactive product will be expected to operate. Four aspects of the environment must be considered when establishing requirements: First is the physical environment such as how much lighting, noise, the second aspect of the environment is the social environment. The third aspect is the organizational environment. Finally, the technical environment.
- User requirements capture the characteristics of the intended user group.
- Usability requirements capture the usability goals and associated measures for a particular

product.

- Usability requirements are related to other kinds of requirement we must establish, such as the kinds of users expected to interact with the product.

5.4. Data gathering

So how do we go about determining requirements? Data gathering is an important part of the requirements activity and also of evaluation.

The purpose of data gathering is to collect sufficient, relevant, and appropriate data so that a set of stable requirements can be produced. Even if a set of initial requirements exists, data gathering will be required to expand, clarify, and confirm those initial requirements. Data gathering needs to cover a wide spectrum of issues because the different kinds of requirement we need to establish are quite varied.

There is essentially a small number of basic techniques for data gathering, but they are flexible and can be combined and extended in many ways. These techniques are:

1) *Questionnaires:* Most of us are familiar with questionnaires. They are a series I of questions designed to elicit specific information from us. The questions may require different kinds of answers: some require a simple YES\NO, others ask us to choose from a set of pre-supplied answers, and others ask for a longer response or comment. Sometimes questionnaires are sent in electronic form and arrive via email or are posted on a website, and sometimes they are given to us on paper. In most cases the questionnaire is administered at a distance, i.e., no one is there to help you answer the questions or to explain what they mean.

2) *Interviews.* Involve asking someone a set of questions. Often interviews are face-to-face, but they don't have to be. Companies spend large amounts of money conducting telephone interviews with their customers finding out what they like or don't like about their service. If interviewed in their own work or home setting, people may find it easier to talk about their activities by showing the interviewer what they do and what systems and other artifacts they use. The context can also trigger them to remember certain things, for example a problem they have downloading email, which they would not have recalled had the interview taken place elsewhere. Interviews can be broadly classified as structured, unstructured or semi structured, depending on how rigorously the interviewer sticks to a prepared set of questions.

3) *Focus groups and workshops.* Interviews tend to be one on one, and elicit only one person's perspective. As an alternative or as corroboration, it can be very revealing to get a group of stakeholders together to discuss issues and requirements. These sessions

can be very structured with set topics for discussion, or can be unstructured. In this latter case, a facilitator is required who can keep the discussion on track and can provide the necessary focus or redirection when appropriate. In some development methods, workshops have become very formalized.

4) *Naturalistic observation.* It can be very difficult for humans to explain what they do or to even describe accurately how they achieve a task. So it is very unlikely that a designer will get a full and true story from stakeholders by using any of the techniques listed above. The scenarios and other props used in interviews and workshops will help prompt people to be more accurate in their descriptions, but observation provides a richer view. Observation involves spending some time with the stakeholders as they go about their day-to-day tasks, observing work as it happens, in its natural setting. A member of the design team shadows a stakeholder, making notes, asking questions (but not too many), and observing what is being done in the natural context of the activity.

5) *Studying documentation.* Procedures and rules are often written down in manuals and these are a good source of data about the steps involved in an activity and only source. Other documentation that might be studied includes diaries or job logs that are written by the stakeholders during the course of their work. In the requirements activity, studying documentation is good for understanding legislation and getting some background information on the work. It also doesn't involve stakeholder time, which is a limiting factor on the other techniques.

Table 5.1 overview of data-gathering techniques used in the requirements activity

Technique	Good for	Kind of data	Advantages	Disadvantages	Detail for designing in
Questionnaires	Answering specific questions	Quantitative and qualitative data	Can reach many people with low resource	The design is crucial. Response rate may be low. Responses may not be what you want	Chapter 13
Interviews	Exploring issues	Some quantitative but mostly qualitative data	Interviewer can guide interviewee if necessary. Encourages contact between developers and users	Time consuming. Artificial environment may intimidate interviewee	Chapter 13
Focus groups and workshops	Collecting multiple viewpoints	Some quantitative but mostly qualitative data	Highlights areas of consensus and conflict. Encourages contact between developers and users	Possibility of dominant characters	Chapter 13
Naturalistic observation	Understanding context of user activity	Qualitative	Observing actual work gives insights that other techniques can't give	Very time consuming. Huge amounts of data	Chapter 12
Studying documentation	Learning about procedures, regulations and standards	Quantitative	No time commitment from users required	Day-to-day working will differ from documented procedures	N/A

5.4.1 Choosing between techniques

Table 5.1 provides some information to help you choose a set of techniques for a specific project. It tells you the kind of information you can get, e.g., answers to specific questions, and the kind of data it yields, e.g., qualitative or quantitative. It also includes some advantages and disadvantages for each technique. The kind of information you want will probably be determined by where you are in the cycle of iterations.

5.5. Data interpretation and analysis

Once the first data-gathering session has been conducted, interpretation and analysis can begin. It's a good idea to start interpretation as soon after the gathering session as possible. The experience will be fresh in the minds of the participants and this can help overcome any bias caused by the recording approach. It is also a good idea to discuss the findings with others to get a variety of perspectives on the data.

The aim of the interpretation is to begin structuring and recording descriptions of requirements. Using a template such as the one suggested in Volere (Figure 5.2) highlights the kinds of information you should be looking for and guides the data interpretation and analysis. Note that many of the entries are concerned with traceability.

Requirement #: Unique *Id*   Requirement Type: **Template** Event/use case #: Origin of
                                                 section   the requirement

Description: **A one-sentence statement of the intention of the requirement**

Rationale: **Why is the requirement considered important or necessary?**

Source: **Who raised this requirement?**

Fit Criterion: **A quantification of the requirement used to determine whether the solution meets the requirement.**

Customer Satisfaction: **Measures the**   Customer Dissatisfaction: **Unhappiness if it is**
        desire to have the requirement   **not implemented**
        implemented

Dependencies: **Other requirements with a change effect**   Conflicts: **Requirements that**
                                                                       contradict this one

Supporting Materials: **Pointer to supporting information**

History: **Origin and changes to the requirement**

Volere
Copyright *O* Atlantic Systems Guild

Figure (5.2) the volere shell for requirements

More focused analysis of the data will follow initial interpretation. Different techniques and notations exist for investigating different aspects of the system that will in turn give rise to the different requirements. For example, functional requirements have traditionally been analyzed and documented using data-flow diagrams,

(a)
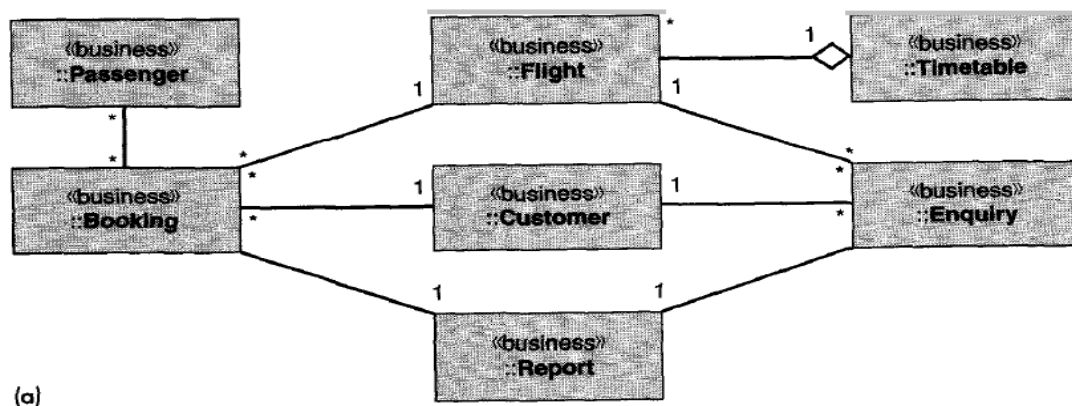
Chapter 6: **Design, Prototyping And Construction 239**

6.1   Introduction 239

6.2   Design, Prototyping and construction 240

    6.2.1   What is design?

6.1. Introduction

Design activities begin once a set of requirements has been established. Broadly speaking, there are two types of design: conceptual and physical. The former is concerned with developing a conceptual model that captures what the product will do and how it will behave, while the latter is concerned with details of the design such as screen and menu structures, icons, and graphics.

The design emerges iteratively, through repeated design-evaluation-redesign cycles involving users. For users to effectively evaluate the design of an interactive product, designers must produce an interactive version of their ideas. in the early stages of development, these interactive versions may be made of paper and cardboard, while as design progresses and ideas become more detailed, they may be polished pieces of software, metal, or plastic that resemble the final product. We have called the activity concerned with building this interactive version prototyping and construction.

There are two distinct circumstances for design: one where you're starting from scratch and one where you're modifying an existing product.

In Chapter 5, we discussed some ways to identify user needs and establish requirements. In this chapter, we look at the activities involved in progressing a set of requirements through the cycles of prototyping to construction.

6.2. Design, Prototyping and construction

6.2.1 What is design

So what is design? A simple definition is:

Achieving goals within constraints

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

Goals What is the purpose of the design we are intending to produce? Who is it for? Why do they want it? For example, if we are designing a wireless personal movie player, we may think about young affluent users wanting to watch the latest movies whilst on the move and download free copies, and perhaps wanting to share the experience with a few friends.

Constraints What materials must we use? What standards must we adopt? How much can it cost? How much time do we have to develop it? Are there health and safety issues? In the case of the personal movie player: does it have to withstand rain? Must we use existing video

standards to download movies? Do we need to build in copyright protection?

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

<u>Trade-off Choosing</u> which goals or constraints can be relaxed so that others can be met. However, the more common skill needed in design is to accept the conflict and choose the most appropriate trade-off.

6.2.1.1 The golden rule of design

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

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

1) Understand computers

2) Limitations, capacities, tools, platforms

3) Understand people

4) Psychological, social aspects, human error.

6.2.1.2 The process of design

Often HCI professionals complain that they are called in too late. A system has been designed and built. In other companies usability is seen as equivalent to testing – checking whether people can use it and fixing problems, In the best companies, however, usability is designed in from the start.

In this section we will look in detail at the software development process and how HCI fits within it. Here we'll take a simplified view of four main phases plus an iteration loop, focused on the design of interaction (Figure below).
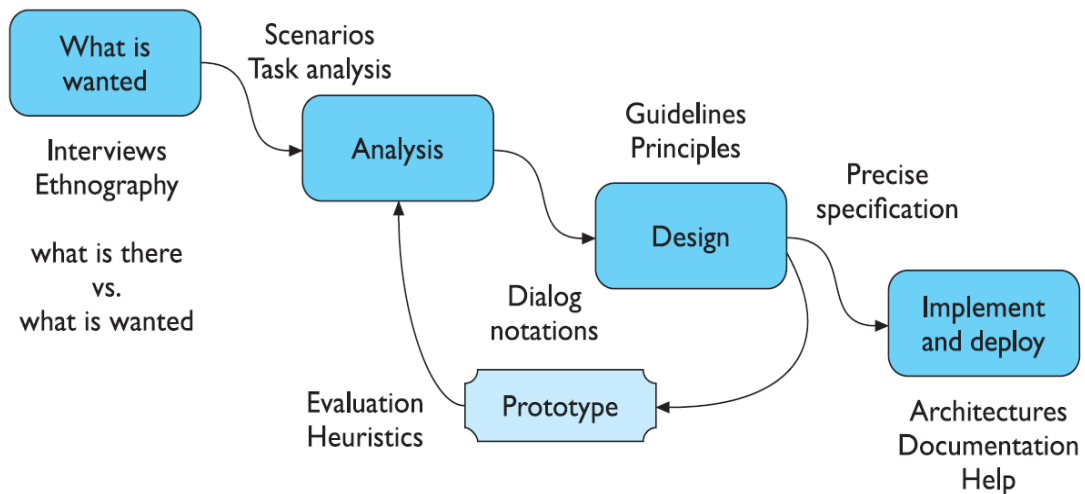
Figure 6.1 interaction design process

Requirements – what is wanted*:* The first stage is establishing what exactly is needed? As a precursor to this it is usually necessary to find out what is currently happening.

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...etc.

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. Which are a means to capture how people carry out the various tasks that are part of their work and life. In this chapter, we will look at scenarios, rich stories of interaction, which can be used in conjunction with a method like task analysis or on their own to record and make vivid actual interaction. These techniques can be used both to represent the situation as it is and also the desired situation.

Design: Well, this is all about design, but there is a central stage when you move from what you want, to how to do it. There are numerous rules, guidelines and design principles that can be used to help us. We need to record our design choices in some way and there are various notations and methods to do this, including those used to record the existing situation. We used this simple notations for designing navigation within a system and some basic heuristics to guide the design of that navigation.

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. Some forms of evaluation can be done using the design on paper, but it is hard to get real feedback without trying it out. Most user interface design therefore involves some form of prototyping, producing early versions of systems to try out with real users.

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.

6.2.2 Prototyping

It is often said that users can't tell you what they want, but when they see something and get to use it, they soon know what they don't want. Having collected information about work practices and views about what a system should and shouldn't do, we then need to try out our ideas by building prototypes and iterating through several versions. And the more iterations, the better the final product will be.

6.2.2.1 What is a prototype

When you hear the term prototype, you may imagine something like a scale model of a building or a bridge, or maybe a piece of software that crashes every few minutes. But a prototype can also be a paper-based outline of a screen or set of screens, an electronic "picture," a video simulation of a task, a three dimensional paper and cardboard mockup of a whole workstation, or a simple stack of hyperlinked screen shots, among other things. In fact, a prototype can be anything from a paper-based storyboard through to a complex piece of software, and from a cardboard mockup to a molded or pressed piece of metal. A prototype allows stakeholders to interact with an envisioned product, to gain some experience of using it in a realistic setting, and to explore imagined uses.

So a prototype is a limited representation of a design that allows users to interact with it and to explore its suitability.

6.2.2.2 Why prototype

- Prototypes are a useful aid when discussing ideas with stakeholders; they are a communication device among team members
- Effective way to test out ideas for yourself.
- The activity of building prototypes encourages reflection in design.

Low-fidelity prototyping: is one that does not look very much like the final product.
For example, it uses materials that are very different from the intended final version, such as paper and cardboard rather than electronic screens and metal.

High-fidelity prototyping: uses materials that you would expect to be in the final product and produces a prototype that looks much more like the final thing.
For example, a prototype of a software system developed in Visual Basic is higher fidelity

than a paper-based mockup; a molded piece of plastic with a dummy keyboard is a higher-fidelity prototype of the "Palm Pilot" than the lump of wood.

6.2.3 Construction: from design to implementation

When the design has been around the iteration cycle enough times to feel confident that it fits requirements, everything that has been learned through the iterated steps of prototyping and evaluation must be integrated to produce the final product.

Although prototypes will have undergone extensive user evaluation, they will not necessarily have been subjected to rigorous quality testing for other characteristics such as robustness and error-free operation. Constructing a product to be used by thousands or millions of people running on various platforms and under a wide range of circumstances requires a different testing regime than producing a quick prototype to answer specific questions.

The dilemma box below discusses two different development philosophies.

❖ One approach, called evolutionary prototyping, involves evolving a prototype into the final product.

❖ An alternative approach, called throwaway prototyping, uses the prototypes as stepping stones towards the final design. In this case, the prototypes are thrown away and the final product is built from scratch. If an evolutionary prototyping approach is to be taken, the prototypes should be subjected to rigorous testing along the way; for throw-away prototyping such testing is not necessary.

Chapter 7: Introducing Evaluation 317

7.1 Introduction

This chapter begins by discussing *what* evaluation is, *why* evaluation is important, and *when* to use different evaluation techniques and approaches. Then a case study is presented about the evaluation techniques used by Microsoft researchers and the Fred Hutchinson Cancer Research Center in developing HutchWorld (Cheng et al., 2000), a virtual world to support cancer patients, their families, and friends. This case study is chosen because it illustrates how a range of techniques is used during the development of a new product. It introduces some of the practical problems that evaluators encounter and shows how iterative product development is informed by a series of evaluation studies. The Hutch World study also lays the foundation for the evaluation framework. The main aims of this chapter are to:

- Explain the key concepts and terms used to discuss evaluation.
- Discuss and critique the HutchWorld case study.
- Examine how different techniques are used at different stages in the development of HutchWorld.
- Show how developers cope with real-world constraints in the development of HutchWorld.

7.2 What, why, and when to evaluate

Users want systems that are easy to learn and to use as well as effective, efficient, safe, and satisfying. Being entertaining, attractive, and challenging, etc. is also essential for some products. *So,* knowing what to evaluate, why it is important, and when to evaluate are key skills for interaction designers.

7.2.1 What to evaluate

There is a huge variety of interactive products with a vast array of features that need to be evaluated. Some features, such as the sequence of links to be followed to find an item on a website, are often best evaluated in a laboratory, since such a setting allows the evaluators to control what they want to investigate. Other aspects, such as whether a collaborative toy is robust and whether children enjoy interacting with it, are better evaluated in natural settings, so that evaluators can see what children do when left to their own devices.

7.2.2 Why you need to evaluate

Just as designers shouldn't assume that everyone is like them, they also shouldn't presume that following design guidelines guarantees good usability, Evaluation is needed to check that users can use the product and like it.

Tognazzi points out that there are five good reasons for investing in user testing:

1. Problems are fixed before the product is shipped, not after.

2. The team can concentrate on real problems, not imaginary ones.

3. Engineers code instead of debating.

4. Time to market is sharply reduced.

5. Finally, upon first release, your sales department has a rock-solid design it can sell without having to pepper their pitches with how it will all actually work in release 1.1 or 2.0.

7.2.3 When to evaluate

The product being developed may be a brand-new product or an upgrade of an existing product. If the product is new, then considerable time is usually invested in market research. Designers often support this process by developing mockups of the potential product that are used to elicit reactions from potential users. As well as helping to assess market need, this activity contributes to understanding users' needs and early requirements uation is to assess how well a design fulfills users' needs and whether users like it.

In the case of an upgrade, there is limited scope for change and attention is focused on improving the overall product. This type of design is well suited to usability engineering in

which evaluations compare user performance and attitudes with those for previous versions. Some products, such as office systems, go through many versions, and successful products may reach double digit version numbers. In contrast, new products do not have previous versions and there may be nothing comparable on the market, so more radical changes are possible if evaluation results indicate a problem.

Evaluations done during design to check that the product continues to meet users' needs are known as *formative evaluations*. Evaluations that are done to assess the success of a finished product, such as those to satisfy a sponsoring agency or to check that a standard is being upheld, are known as *summative evaluation*. Agencies such as National Institute of Standards and Technology (NIST) in the USA, the International Standards Organization (ISO) and the British Standards Institute (BSI) set standards by which products produced by others are evaluated.

7.3 HutchWorld case study

HutchWorld is a distributed virtual community developed through collaboration between Microsoft's Virtual Worlds Research Group and librarians and clinicians at the Fred Hutchinson Cancer Research Center in Seattle, Washington. The system enables cancer patients, their caregivers, family, and friends to chat with one another, tell their stories, discuss their experiences and coping strategies, and gain emotional and practical support from one another (Cheng et. al.,2000). The design team decided to focus on this particular population because caregivers and cancer patients are socially isolated: cancer patients must often avoid physical contact with others because their treatments suppress their immune systems. Similarly, their caregivers have to be careful not to transmit infections to patients.

7.3.1   How the design team got started: early design ideas

Before developing this product, the team needed to learn about the patient experience at the Fred Hutchinson Center. For instance, what is the typical treatment process, what resources are available to the patient community, and what are the

needs of the different user groups within this community? They had to be particularly careful about doing this because many patients were very sick. Cancer patients also typically go through bouts of low emotional and physical energy.

Caregivers also may have difficult emotional times, including depression, exhaustion, and stress. Furthermore, users vary along other dimensions, such as education and experience with computers, age and gender and they come from different cultural backgrounds with different expectations.

The development team decided that HutchWorld should be available for patients any time of day or night, regardless of their geographical location.The team's informal visits to the Fred Hutchinson Center led to the development of an early prototype. They followed a user-centered development methodology. Having got a good feel for the users' needs, the team brainstormed different ideas for an organizing theme to shape the conceptual design a conceptual model possibly based on a metaphor. After much discussion, they decided to make the design resemble the outpatient clinic lobby of the Fred Hutchinson Cancer ResearchCenter. By using this real-world metaphor, they hoped that the users would easily infer what functionality was available in HutchWorld from their knowledge of the real clinic. The next step was to decide upon the kind of communication environment to use. Should it be synchronous or asynchronous? Which would support social and affective communications best? **A** synchronous chat environment was selected because the team thought that this would be more realistic and personal than an asynchronous environment. They also decided to include 3D photographic avatars so that users could enjoy having an identifiable online presence and could easily recognize each other.

The prototype was reviewed with users throughout early development and was later

tested more rigorously in the real environment of the Hutch Center using a variety of

techniques.

A Microsoft product called V-Chat was used to develop second interactive prototype with the subset of the features in the preliminary design, however, only the lobby was fully developed. Before testing could begin, the team had to solve some logistical issues. There were two key questions. Who would provide training for the testers and help for the patients? And how many systems were needed for testing and where should they be placed? As in many high - tech companies, the Microsoft team was used to short, market-driven production schedules, but this time they were in for a shock.

Organizing the testing took much longer than they anticipated, but they soon learned to set realistic expectations that were in synch with hospital activity and the unexpected delays that occur when working with people who are unwell.

1. **How was the testing done?**

   The team ran two main sets of user tests. The first set of tests was informally run onsite at the Fred Hutchinson Center in the hospital setting. After observing the system in use on computers located in the hospital setting, the team redesigned the software and then ran formal usability tests in the usability labs at Microsoft.

Test 1 : Early observations onsite

In the informal test at the hospital, six computers were set up and maintained by Hutch staff members. A simple, scaled -back prototype of HutchWorld was built using the existing product, Microsoft V-Chat and was installed on the computers,

which patients and their families from various hospital locations used. Over the course of several months, the team trained Hutch volunteers and hosted events in

the V-Chat prototype. The team observed the usage of the space during unscheduled times, and they also observed the general usage of the prototype.

Test 1 : What was learned?

This V-Chat test brought up major usability issues. First, the user community was relatively small, and there were never enough participants in the chat room for successful communication-a concept known as ***critical mass.*** In addition, many of the patients were not interested in or simultaneously available for chatting. Instead, they preferred asynchronous communication, which does not require an immediate response. Patients and their families used the computers for email, journals, discussion lists, and the bulletin boards largely because they could be used at any time and did not require others to be present at the same time. The team learned that a strong asynchronous base was essential for communication. The team also observed that the users used the computers to play games and to search the web for cancer sites approved by Hutch clinicians. This information was not included in the virtual environment, and so users were forced to use many different applications. **A** more "unified" place to find all of the Hutch content was desired that let users rapidly swap among a variety of communication, information, and entertainment tasks.

Test 1 : The redesign

Based on this trial, the team redesigned the software to support more asynchronous communication and to include a variety of communication, information, and entertainment areas. They did this by making HutchWorld function as a portal that provides access to information -retrieval tools, communication tools, games, and other types of entertainment. Other features were incorporated too, including email, a bulletin board, a text-chat, a web page creation tool, and a way of checking to see if anyone is around to chat with in the 3D world.

Test 2: Usability tests

After redesigning the software, the team then ran usability tests in the Microsoft

usability labs. Seven participants (four male and three female) were tested. Four of these participants had used chat rooms before and three were regular users. All had browsed the web and some used other communications software. The participants were told that they would use a program called HutchWorld that was designed to provide support for patients and their families. They were then given five minutes to explore HutchWorld. They worked independently and while they explored they provided a running commentary on what they were looking at, what they were thinking, and what they found confusing. This commentary was recorded on video and so were the screens that they visited, so that the Microsoft evaluator, who watched through a one -way mirror, had a record of what happened for later analysis. Participants and the evaluator interacted via a microphone and speakers. When the five-minute exploration period ended, the participants were asked to complete a series of *structured tasks* that were designed to test particular features of the HutchWorld interface. These tasks focused on how participants dealt with their virtual identity; that is, how they represented themselves and were perceived by others communicated with others got the information they wanted found entertainment

2.   Was it tested again?

Following the usability testing, there were more rounds of observation and testing with six new participants, two males and four females. These tests followed the same general format as those just described but this time they tested multiple users at once, to ensure that the virtual world supported multiuser interactions. The tests were also more detailed and focused. This time the results were more positive, but of course there were still usability problems to be fixed. Then the question arose: what to do next? In particular, had they done enough testing (see Dilemma)?

After making a few more fixes, the team stopped usability testing with specific tasks. But the story didn't end here. The next step was to show HutchWorld to cancer patients and caregivers in a focus-group setting at the Fred Hutchinson Cancer Research Center to get their feedback on the final version. Once the team made adjustments to HutchWorld in response to the focus-group feedback, the final step was to see how well HutchWorld worked in a real clinical environment. It was therefore taken to a residential building used for long term patient and family stays that was fully wired for Internet access. Here, the team observed what happened when it was used in this natural setting. In particular, they wanted to find out how HutchWorld would integrate with other aspects of patients' lives, particularly with their medical care routines and their access to social support. This informal observation allowed them to examine patterns of use and to see who used which parts of the system, when, and why.

3.   Looking *to* the future

Future studies were planned to evaluate the effects of the computers and the software in the Fred Hutchinson Center. The focus of these studies will be the social support and wellbeing of patients and their caregivers in two different conditions. There will be a control condition in which users (i.e., patients) live in the residential building without computers and an experimental condition in which users live in similar conditions but with computers, Internet access, and HutchWorld. The team will evaluate the user data (performance and observation) and surveys collected in the study to investigate key questions, including:

1- How does the computer and software impact the social wellbeing of patients and their caregivers?
2- What type of computer-based communication best supports this patient community?
3- What are the general usage patterns? i.e., which features were used and at
4- What time of day were they used, etc.?
5- How might any medical facility use computers and software like Hutch-World to provide social support for its patients and caregivers?

7.4   Evaluation paradigms and techniques

Before we describe the techniques used in evaluation studies, we shall start by proposing some key terms. We start with the much-used term user studies, defined by Abigail Sellen as follows: "user studies essentially involve looking at how people behave either in their natural environments, or in the laboratory, both with old technologies and with new ones." Any kind of evaluation, whether it is a user study or not, is guided either explicitly or implicitly by a set of beliefs that may also be underpinned by theory. These beliefs and the practices (i.e., the methods or techniques) associated with them are known as an evaluation paradigm, which you should not confuse with the "interaction paradigms" discussed in Chapter 2. Often evaluation paradigms are related to a particular discipline in that they strongly influence how people from the discipline think about evaluation.

Each paradigm has particular methods and techniques associated with it. We tend to talk about techniques, but you may find that other books call them methods. An example of the relationship between a paradigm and the techniques used by evaluators following that paradigm can be seen for usability testing, which is an applied science and engineering paradigm. The techniques associated with usability testing are: user testing in a controlled environment; observation of user activity in the controlled environment and the field; and questionnaires and interviews.

7.4.1 Evaluation Paradigms

In this part we identify four core evaluation paradigms:

1)) A "quick and dirty" evaluation is a common practice in which designers informally get feedback from users or consultants to confirm that their ideas are in line with users' needs and are liked. "Quick and dirty" evaluations can be done at any stage and the emphasis is on fast input rather than carefully documented findings. For example, early in design developers may meet informally with users to get feedback on ideas for a new product (Hughes et al., 1994). At later stages similar meetings may occur to try out an idea for an icon, check whether a graphic is liked, or confirm that information has been appropriately categorized on a webpage. This approach is often called "quick and dirty" because it is meant to be done in a short space of time. Getting this kind of feedback is an essential ingredient of successful design.

2)) Usability testing: was the dominant approach in the 1980s, and remains important, although, as you will see, field studies and heuristic evaluations have grown in prominence. Usability testing involves measuring typical users' performance on carefully prepared tasks that are typical of those for which the system was designed. Users' performance is generally measured in terms of number of errors and time to complete the task. As the users perform these tasks, they are watched and recorded on video and by logging their interactions with software. This observational data is used to calculate performance times, identify errors, and help explain why the users did what they did. User satisfaction questionnaires and interviews are also used to elicit users' opinions.

3)) Field studies: The distinguishing feature of field studies is that they are done in natural settings with the aim of increasing understanding about what users do naturally and how technology impacts them. In product design, field studies can be used to :

1- help identify opportunities for new technology
2- determine requirements for design
3- facilitate the introduction of technology
4- Evaluate technology.

4)) Predictive evaluation: In predictive evaluations experts apply their knowledge of typical users, often guided by heuristics, to predict usability problems. Another approach involves theoretically based models. The key feature of predictive evaluation is that users need not be present, which makes the process quick, relatively inexpensive, and thus attractive to companies; but it has limitations.

Table 7.1 characteristics of different evaluation paradigms

Evaluation paradigms	"Quick and dirty"	Usability testing	Field studies	Predictive
Role of users	Natural behavior.	To carry out set tasks.	Natural behavior.	Users generally not involved.
Who controls	Evaluators take minimum control.	Evaluators strongly in control.	Evaluators try to develop relationships with users.	Expert evaluators.
Location	Natural environment or laboratory.	Laboratory.	Natural environment.	Laboratory-oriented but often happens on customer's premises.
When used	Any time you want to get feedback about a design quickly. Techniques from other evaluation paradigms can be used–e.g., experts review software.	With a prototype or product.	Most often used early in design to check that users' needs are being met or to assess problems or design opportunities.	Expert reviews (often done by consultants) with a prototype, but can occur at any time. Models are used to assess specific aspects of a potential design.
Type of data	Usually qualitative, informal descriptions.	Quantitative. Sometimes statistically validated. Users' opinions collected by questionnaire or interview.	Qualitative descriptions often accompanied with sketches, scenarios, quotes, other artifacts.	List of problems from expert reviews. Quantitative figures from model, e.g., how long it takes to perform a task using two designs.
Fed back into design by . . .	Sketches, quotes, descriptive report.	Report of performance measures, errors etc. Findings provide a benchmark for future versions.	Descriptions that include quotes, sketches, anecdotes, and sometimes time logs.	Reviewers provide a list of problems, often with suggested solutions. Times calculated from models are given to designers.
Philosophy	User-centered, highly practical approach.	Applied approach based on experimentation, i.e., usability engineering.	May be objective observation or ethnographic.	Practical heuristics and practitioner expertise underpin expert reviews. Theory underpins models.

7.4.2 Techniques

There are many evaluation techniques and they can be categorized in various ways, but in this text we will examine techniques for:

1. observing users
2. asking users their opinions
3. asking experts their opinions
4. testing users' performance
5. modeling users' task performance to predict the efficacy of a user interface

The brief descriptions below offer an overview of each category, Be aware that some techniques are used in different ways in different evaluation paradigms.

1- Observing users: Observation techniques help to identify needs leading to new types of products and help to evaluate prototypes. Notes, audio, video, and interaction logs are well known ways of recording observations and each has benefits and drawbacks. Obvious challenges for evaluators are how to observe without disturbing the people being observed and how to analyze the data, particularly when large quantities of video data are collected or when several different types must be integrated to tell the story (e.g., notes, pictures, and sketches from observers).

2- Asking users: what they think of a product-whether it does what they want; whether they like it; whether the aesthetic design appeals; whether they had problems using it; whether they want to use it again-is an obvious way of getting feedback. Interviews and questionnaires are the main techniques for doing this. The questions asked can be unstructured or tightly structured. They can be asked of a few people or of hundreds. Interview and questionnaire techniques are also being developed for use with email and the web.

3- Asking experts: Software inspections and reviews are long established techniques for evaluating software code and structure. During the 1980s versions of similar techniques were developed for evaluating usability. Guided by heuristics, experts step through tasks role-playing typical users and identify problems. Developers like this approach because it is usually relatively inexpensive and quick to perform compared with laboratory and field evaluations that involve users. In addition, experts frequently suggest solutions to problems.

4- User testing: Measuring user performance to compare two or more designs has been the bedrock of usability testing. As we said earlier when discussing usability testing, these tests are usually conducted in controlled settings and involve typical users performing typical, well-defined tasks. Data is collected so that performance can be analyzed. Generally the time taken to complete a task, the number of errors made, and the navigation path through the product are recorded. Descriptive statistical measures such as means and standard deviations are commonly used to report the results.

5- Modeling users' task performance: There have been various attempts to model human-computer interaction so as to predict the efficiency and problems associated with different designs at an early stage without building elaborate prototypes. These techniques are successful for systems with limited functionality such as telephone systems are the best known techniques.

Table 7.2 below summarizes the categories of techniques and indicates how they are commonly used in the four evaluation paradigms.

	Evaluation paradigms			
Techniques	"Quick and dirty"	Usability testing	Field studies	Predictive
Observing users	Important for seeing how users behave in their natural environments.	Video and interaction logging, which can be analyzed to identify errors, investigate routes through the software, or calculate performance time.	Observation is the central part of any field study. In ethnographic studies evaluators immerse themselves in the environment. In other types of studies the evaluator looks on objectively.	N/A
Asking users	Discussions with users and potential users individually, in groups or focus groups.	User satisfaction questionnaires are administered to collect users' opinions. Interviews may also be used to get more details.	The evaluator may interview or discuss what she sees with participants. Ethnographic interviews are used in ethnographic studies.	N/A
Asking experts	To provide critiques (called "crit reports") of the usability of a prototype.	N/A	N/A	Experts use heuristics early in design to predict the efficacy of an interface.
User testing	N/A	Testing typical users on typical tasks in a controlled laboratory-like setting is the cornerstone of usability testing.	N/A	N/A
Modeling users' task performance	N/A	N/A	N/A	Models are used to predict the efficacy of an interface or compare performance times between versions.

7.5 DECIDES: A framework to guide evaluation

Well-planned evaluations are driven by clear goals and appropriate questions. To guide our evaluations we use the DECIDE framework, which provides the following checklist to help novice evaluators:

1. Determine the overall goals that the evaluation addresses.

2. Explore the specific questions to be answered.

3. Choose the evaluation paradigm and techniques to answer the questions.

4. Identify the practical issues that must be addressed, such as selecting participants.

5. Decide how to deal with the ethical issues.

6. Evaluate, interpret, and present the data.

7.6 Discussion

In both HutchWorld and the 1984 Olympic Messaging System, a variety of evaluation techniques were used at different stages of design to answer different questions.

"Quick and dirty" observation, in which the evaluators informally examine how a prototype is used in the natural environment, was very useful in early design. Following this with rounds of usability testing and redesign revealed important usability problems. However, usability testing alone is not sufficient.

Field studies were needed to see how users used the system in their natural environments, and sometimes the results were surprising. For example, in the OMS system users from

different cultures behaved differently. A key issue in the HutchWorld study was how use of the system would fit with patients' medical routines and changes in their physical and emotional states. Users' opinions also offered valuable insights. After all, if users don't like a system, it doesn't matter how successful the usability testing is: they probably won't use it. Questionnaires and interviews were used to collect user's opinions.

An interesting point concerns not only how the different techniques can be used to address different issues at different stages of design, but also how these techniques complement each other. Together they provide a broad picture of the system's usability and reveal different perspectives. In addition, some techniques are better than others for getting around practical problems. This is a large part of being a successful evaluator. In the HutchWorld study, for example, there were not many users, so the evaluators needed to involve them sparingly. For example, a technique requiring 20 users to be available at the same time was not feasible in the HutchWorld study, whereas there was no problem with such an approach in the OMS study. Furthermore, the OMS study illustrated how many different techniques, some of which were highly opportunistic, can be brought into play depending on circumstances. Some practical issues that evaluators routinely have to address include:

1. what to do when there are not many users
2. how to observe users in their natural location (i.e., field studies) without disturbing them
3. having appropriate equipment available
4. dealing with short schedules and low budgets
5. not disturbing users or causing them duress or doing anything unethical
6. collecting "useful" data and being able to analyze it
7. selecting techniques that match the evaluators' expertise