

Chapter 2

Models of the user interface design process

Chapter summary

At present there is no one standard approach to designing user interfaces. This is because the field of user interface design has developed from many disciplines, each of which has contributed its own perspective and practices (see Chapter 1). Several different models of the process exist, each of which employs its own combination of design techniques. Specific design techniques are covered in Chapter 5.

This chapter introduces you to some of the key process models for creating user interface designs.

Learning outcomes

Outcome 1: The design lifecycle – understand the stages in the software lifecycle and where user interface design considerations fit in.

Outcome 2: Approaches to user interface design – identify different perspectives of user interface design.

Outcome 3: The development team – understand the roles within the software development team.

Outcome 4: Participative design (PD) – understand some specific user interface design models within the PD approach.

Question 1 at the end of the chapter tests you on this.

Outcome 5: User involvement – understand the advantages and difficulties of involving users in the user interface design process.

Question 2 at the end of the chapter tests you on this.

Outcome 6: Prototyping – understand the benefits and limitations of prototyping user interfaces.

How will you be assessed on this?

You will be expected to be aware of the relationship between user interface design and the wider software development process. This includes where user interface considerations fit in the software development life cycle, the different approaches to user interface design identified by Wallace and Anderson (1993) and the range and nature of roles within the software development team. You will need to explain certain approaches to user interface design within the participative design tradition and be able to describe and discuss the nature of prototyping user interfaces.

Section 1

The design lifecycle

The process of developing software can be regarded as a stage model. Certain activities happen at each stage and each stage has to be completed before the next can occur. A key consideration for user interface designers is where, in the overall software development process, the design of the user interface takes place.

A traditional model of systems development is the system lifecycle. There are many variants of this model, such as the Waterfall model (see Benyon, 1995), but a generic system lifecycle might look like this:

- requirements and function analysis;
- preliminary, high-level design;
- specification;
- design;
- testing and evaluation;
- production;
- maintenance.

CRUCIAL CONCEPT

The software development process is often considered to be **cyclical**: various activities take place in sequence. At the end of the process there is often a need for the product to be updated or more functionality to be added, so the cycle starts again.

At any stage, there might be a degree of iteration to the previous stage; and at the end of the cycle, there might be another complete cycle (a new version of the product is required, for instance).

CRUCIAL CONCEPT

Iteration, in this context, means going back a step in the cycle to deal with a problem that has come to light.

The point of interest from the perspective of interface designers is where in the cycle the design of the interface takes place.

Traditionally, the interface tended to be built towards the end of the cycle, when most of the functionality had been programmed in. There has been a growing awareness that more attention needs to be directed towards the design of the interface, especially since modern applications have a substantial proportion of the overall code supporting the interface; and because the majority of users are not experts. There now exist a number of approaches that explicitly address considerations of the design of the interface at points throughout the design lifecycle.

Quick test

1. Should the design of the user interface be left until the end of the software development process?
2. What does iteration mean?

Section 2

Approaches to interface design

This section looks at an influential paper by Wallace and Anderson (1993) that describes four discrete approaches to the design of user interfaces and suggests some characteristics that an ideal approach should possess.

Wallace and Anderson (1993) suggest that there are four identifiable approaches to user interface design:

- the craft approach;
- cognitive engineering;
- enhanced software engineering;
- the technologist approach.

The **craft approach** produces designs on the basis of the skill and experience of an expert designer who can be seen as a craftsman or artist. Each new design is unique and relies on the talent and creative abilities of the master designer. Design expertise is communicated by a process similar to that between a master craftsman and an apprentice: the apprentice learns by observation, imitation and practice. There is often no explicit process: the design is driven by the implicit knowledge of the expert.

This approach was common in the early days of user interface design and is still evident today, particularly with certain website designs. However, it tends to become inefficient once the problem domain becomes very complex and when changes in technologies occur. A good example of this was the development of graphical user interfaces in the middle 1980s: expert designers of command line interfaces were confronted with a transformation of the technology and much of their implicit knowledge was no longer applicable.

CRUCIAL CONCEPT

Craft approaches to user interface design tend to become inefficient when the design context changes or becomes very complex.

Cognitive engineering involves the application of cognitive psychological theory to user interface design. Cognitive psychology is broadly the study of how people think and, in particular, how they process information in order to guide action. Various models, such as the GOMS family of models (see Card, Moran and Newell, 1983) have been developed to predict how humans will interact with various alternative designs. The attraction of this approach is that applying these models to designs in their early stages (such as paper-based mock-ups) designers can identify optimum designs before embarking on the often time-consuming activity of producing fully functional applications (see Chapter 6 for a discussion of the use of predictive models in the evaluation of early designs). However, the early promise of cognitive engineering has not been widely fulfilled because it has proved time-consuming for most user interface designers to understand and learn to apply the models and also because the models themselves tend to have a very narrow scope of application.

Enhanced software engineering involves the introduction of HCI techniques into established software development methodologies. Structured design methods have not primarily been concerned with design of the user interface. Some successful attempts have been made to embed user interface techniques into, for example, SSADM and JSD (see Silcock *et al.*, 1990), in order to produce a specification for the user interface as well as for the underlying functional code. Again, however, the level of expertise required to carry out

such analyses and the sheer complexity of the task has meant that, in practice, enhanced software engineering has not been widely adopted.

The **technologist approach** aims at automating the design process through the use of software tools, such as **user interface management systems (UIMS)**. These tools support designers in specifying the layout or screen design, together with the dialogue between user and application.

Wallace and Anderson suggest that the aim of each of these approaches should be to improve the quality of both the completed design and the design process. They are all deficient in some respects, however, and the authors propose a set of 13 criteria for an effective design process. These include, for example:

- flexibility – any such approach should be adaptive to the needs of its users;
- communication – an ideal approach must enhance communication between those involved in the process;
- productivity – designer productivity must demonstrably improve;
- user participation – the process should support the participation of end users in the process;
- ease of use – the process should be easy to use and to learn;
- scope – the process should cover the whole of the product life cycle.

To date, no single user interface design process has been developed which incorporates all 13 criteria. Certain approaches within the participative design tradition address some of these criteria, particularly emphasising user involvement and communication between those involved in the process. Section 4 in this chapter covers participative design.

Quick test

1. What are the main weaknesses of the four approaches to user interface design described by Wallace and Anderson (1993)?
2. What characteristics should an ideal user interface design approach possess?

Section 3

The development team

This section looks at certain socio-organisational issues of product development teams, such as roles, responsibilities and structures. It also looks at problems that can arise and the arrangements that can be put in place to resolve them.

The design of user interfaces clearly cannot take place in isolation from the broader product development process. Some years ago it was not unknown for the user interface to be produced once the underlying application code had been developed: effectively an 'add-on'. It is now generally accepted that the user interface is too important to be regarded in this manner and that the development of the user interface must proceed in parallel with that of the functional software. In many applications nowadays the proportion of the code directly driving interaction with the user is greater than 50%.

This change in orientation has implications for the organisation of the product development team. Should it be organised as separate functional groups such as software engineers, user interface designers and usability testers; or as fully integrated, cross-functional groups? There is some evidence (e.g. Allen, 1995; Hutchings and Knox, 1995) that having different specialists working closely together in teams has significant benefits. Software engineers develop greater awareness of the importance of good user interface

design and the requirements of end-users of their products, whilst usability specialists gain understanding of the software development process.

It is important that, within large application development projects where several designers may be contributing to the design of the user interface, strategic decisions are made by a single expert. This notion is advocated by authors such as Foley (1983) who suggests that a user interface architect take overall control for the structure and behaviour of the user interface. This view is echoed by Gugerty (1993) who uses the term 'superdesigner'. If such a role is not identified there is a real danger that the resulting product will be inconsistent in appearance and behaviour.

Whatever the particular organisational model adopted, problems and conflicts often arise within development teams due to misunderstandings, inadequate communication, internal politics etc. Whilst it is ultimately the responsibility of the manager of the team to resolve these problems, certain imperatives stand out. In the experience of the authors, it is especially important that:

- design decisions are clearly recorded and confirmed;
- responsibilities are firmly established;
- clear channels of communication are put in place.

A further imperative, emphasised by Kraut and Streeter (1995) is that the co-ordination of activities of team members is effectively managed. On large projects, particularly where there is a substantial degree of end-user involvement in the process, or when there is a significant degree of formative evaluation (see Chapter 6) and iterative design taking place, certain members of the design team may not be able to continue with their work until they receive inputs from others. This is not only inefficient but can be dangerous: rather than wait for the results of end-user feedback, designers may become impatient and produce further work along the wrong lines.

Quick test

1. What are the responsibilities of a user interface architect?
2. What kinds of problems can arise within development teams?

Section 4

Participative design

Participative design (PD) is a design philosophy that emphasises the importance of the end-users as key stakeholders in the software development process. Certain examples of PD models and processes are described and some problems of user involvement are discussed.

A prominent approach to product design in general and user interface design in particular is that of participative design (sometimes called participatory design). PD is a design philosophy or movement that arose in the Scandinavian countries and has become influential in Northern America and Western Europe in recent years. The central tenet of PD is that end-users of products are central actors in the development process. Most non-PD developmental approaches acknowledge user-involvement to some extent; in PD, however, users are regarded as integral to the process.

CRUCIAL CONCEPT

User implies anyone who has some connection with the completed product. **End user** refers more specifically to those users who will use the product to a significant degree in their working lives.

PD embraces a range of developmental practices, methodologies, techniques and tools. It also incorporates many different models of how users participate in the development process.

Models of interaction between developers and users

For some development teams, the primary challenge is to acknowledge that the needs of users or customers should be taken seriously. A study by Allen (1995) describes the situation in a prominent software organisation where the products, although fully functional, were not being used by customers. Allen ascribes this to a culture prevailing within the team whereby the developers believed that they knew what was best for users and the emphasis was on writing elegant code rather than creating usable products. Allen describes how he, as the newly appointed manager of this team managed, over time, to establish a customer-centred paradigm within the team.

CRUCIAL CONCEPT

Paradigm – in this context a framework or mind-set – is where the needs of customers come first.

Other models emphasise more strongly the need for more direct user participation in the development process. **Joint application development (JAD)** is now a well-established method where users participate in design teams (see e.g. Bødker, 1996).

Contextual design (Beyer and Holzblatt, 1995) differs from JAD in that the designers do a significant amount of their work in the end-users' own environments. A project starts by the designers talking to and observing end-users as they do their everyday work. The data that emerge from these sessions is then interpreted within cross-functional teams so that all involved can gain a shared perspective on the users' world. Solutions are progressively and iteratively developed using storyboards and paper-based mock-ups, until the final design is agreed upon in the form of an object model. Only then does the software implementation begin.

User-centred design approaches

PD approaches have started to influence the software development practices of a number of prominent organisations. Several companies have produced their own proprietary design approaches, based on the notion of the centrality of the end-user in the process. Some examples include:

- IBM's UCD (Vredenburg, 1999)
- Bang and Olufsen (Bærentsen and Slavensky, 1999)
- Kommunedata (KMD) (Gardner, 1999)
- Microsoft (Muller and Czerwinski, 1999)

As an illustration of these user-centred approaches, the **user-involvement method (UIM)** (see Axtell et al., 1997) is here outlined in Figure 2.1. This method was developed in a large UK-based organisation in order to help with the design and development of a substantial software application to support the administrative activities of a large number of staff. Following a lengthy requirements analysis phase a user group was established, which consisted of four or five representative users. Alongside the user group was a cooperative of one user and a developer.

On the basis of information in the requirements analysis, the cooperative first undertook a task analysis (see Chapter 5 for a description of task analysis). The accuracy of the task analysis was checked with the user group. The cooperative then produced prototype screen designs, which were also given to the user group for their comment and feedback.

In this way, aspects of the design could be successively built up, refined and validated. This iterative cycle of design and feedback continued until the evolving design was sufficiently stable to be implemented.

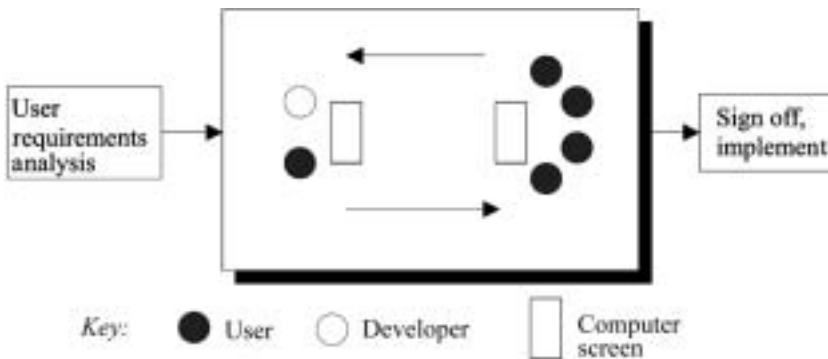


Figure 2.1 Illustration of UIM (after Axtell *et al.*, 1997)

Who are the users?

A key issue in PD-influenced methodologies is who actually are the users of the product. This might seem a perverse question and clearly the answer is that, ideally, those end users who will actually use the product must be involved with the development team. Instead, in many cases, there is a marked tendency for intermediaries or surrogates to take the role of end users, e.g.

- marketing people;
- systems analysts;
- supervisors etc.

Keil and Carmel (1995) report an investigation of 14 software development projects, half of which were regarded as successful whilst the remainder failed in some respects to achieve their aims. In particular, the authors studied the types of links (communication channels) between developers and users within each of these projects. Links included, for example, JAD workshops, focus groups and exposure of users to prototypes. The findings were that the successful projects had used **more links** and, significantly, **more direct links** than unsuccessful projects. By direct links, the authors mean that designers were able to interact directly with the end-users rather than relying on intermediaries (such as supervisors, middle managers) to communicate requirements.

Participative design techniques

PD is both eclectic and pragmatic in its adoption and use of techniques. Muller *et al.* (1993) provide a useful classification of commonly used techniques, based on the point in the development process they are most appropriately used, and whether the emphasis is on the user entering the designers' world or vice versa.

CRUCIAL TIP

PD is eclectic in the sense that techniques are taken from a range of different disciplines and pragmatic because the focus is on practicality and efficiency.

Is user involvement always beneficial?

PD methodologies clearly meet several of Wallace and Anderson's criteria for an ideal process (see Section 2 in this chapter) such as a high degree of user-involvement in the

process, but does it meet them all? A study by Heinbokel *et al.* (1996) suggests that there are problems with user involvement in the development process. For example this particular study demonstrated that user participation led to:

- low overall success of the project;
- few design innovations;
- high stress in the team.

The study by Axtell *et al.* (1997), outlined above, revealed problems with the particular model of user-involvement adopted (UIM), such as:

- too much repetition of work, confusion over decisions;
- time for design decisions to be taken was too long;
- users don't possess sufficient knowledge to contribute significantly;
- users require significant training.

It is, of course, unrealistic to assume that, by bringing end-users to the centre of the product development process, all problems will be automatically resolved. Large-scale software development is an extraordinarily complex undertaking, requiring high levels of technical and managerial competence in a context of changing organisational and human requirements. The issue remains, however, that if end-users are ignored or marginalised in the process, there are very real risks that software products will be underused or rejected by the very people for whom they have been designed.

A study by Damodaran (1996) discusses some of the problems associated with user involvement. One interesting phenomenon that she discusses is the tendency of some users involved with the design team to become 'hostages'. Because of their relative lack of technical knowledge such users tend to go along with decisions made by the designers in order to keep the peace. Consequently design decisions are unchallenged and user-involvement is illusory.

Damodaran provides practical advice regarding such issues as identifying the range of user roles (top management, middle management and end-users), the training needs of users and how the process of user involvement should be managed.

Quick test

1. What are the benefits and problems associated with a high degree of user-involvement?
2. What evidence is there that direct links between developers and end-users result in more successful projects?

Section 5

Prototyping

In this section, the use of **prototyping** as a means of creating user interfaces is presented. Various categories of prototypes are outlined and their advantages and limitations are discussed.

Prototyping is the activity of creating partial designs, quickly and at low cost, in order to allow designers to get feedback from users at an early stage of the design process. It can also be used to create several alternative designs which can be compared or evaluated using one or more **formative techniques** (see Chapter 6). Prototyping is commonly used

within the participative design tradition. Its benefits are widely accepted but there is no single, standardised process.

The prototyping process

Although there are many variations in the practice of prototyping, the following is a summary of the generic process:

- (a) **identify initial user requirements** using techniques such as focus groups, stakeholder analysis etc.;
- (b) **develop a prototype** using appropriate screen layout, dialogue design and navigational guidelines;
- (c) **use and evaluate the prototype** with the aid of user tests, walkthroughs etc.;
- (d) **revise the prototype** using the evaluation results;
- (e) **go to** (c).

How many times it is necessary to go around the loop between (c) and (e) depends on a number of factors such as how quickly usability faults are discovered and how much time is available.

Types of prototypes

There are several different species of prototypes. Rettig (1994) makes the distinction between 'hi-tech' and 'low-tech' while other practitioners (e.g. Greenberg, 1998) use a categorisation of:

- low-fidelity;
- medium-fidelity;
- high-fidelity prototypes.

Broadly, low-tech or low-fidelity prototyping involves the creation of paper-based sketches or mock-ups to represent the appearance and behaviour of the user interface. High-fidelity prototypes are substantially complete computer-based applications, which allow users to interact with much of the functionality of the final product. Medium-fidelity prototypes typically are computer-based simulations, which may not be fully interactive and which allow access to only a small degree of functionality.

CRUCIAL CONCEPT

The **fidelity** of prototypes refers to the extent to which the prototype design accurately reflects the appearance (e.g. the screen layout) and behaviour of the application.

Another distinction that is made is between horizontal and vertical prototypes (see Rudd *et al.*, 1996). Vertical prototypes are high-fidelity prototypes which, however, represent only a proportion of the intended functionality of the final product. Horizontal prototypes model the complete range of user interaction at a surface level but do not provide access to the detailed, lower-level functionality.

Low-fidelity prototypes

Various paper-based tools are used to mock-up sketches, scenarios and storyboards (see Chapter 5 for a description of these design techniques). These tools can range from hand-drawn representations of screen layouts on paper to more elaborate models created from kits of user interface components, such as cardboard buttons, menu bars, dialogue boxes, scroll bars etc.

Once the prototype has been created, users are encouraged to interact with it. Rettig (1994) describes a process for obtaining quick feedback from users using low-tech prototypes. One member of the design team acts as a facilitator, giving users instructions and advice while another team member acts as the 'computer', implementing the

consequences of user actions by manipulating the user interface components in response to user actions. Other team members videotape the session or take notes. At the end of the session, the team discusses the findings and any changes to the prototype can be easily made.

Medium- and high-fidelity prototypes

There exists a range of software prototyping tools, such as HyperCard, Microsoft PowerPoint and Microsoft Visual Basic, which permit partial or fully functional prototypes to be relatively quickly and easily created. Users can interact more naturally with computer-based than with paper-based prototypes: selection of an icon, for example, will cause the appropriate function to be fired or, at least, simulated.

The higher the fidelity of the prototype, the more realistic it will appear to users. In particular, it is easier to design and test the dynamic (i.e. navigational) aspects of the user interface using computer-based prototypes rather than paper-based ones.

Selection of prototyping approaches

Low-fidelity prototyping is generally regarded as a quick and low-cost activity that provides useful feedback about the overall design concept and static (i.e. screen layout) aspects of the user interface design. Higher-fidelity prototypes tend to take longer to develop and require a degree of expertise in using a software prototyping tool. The selection of which approach to take depends on a range of factors, such as:

- the size and scope of the intended application;
- time frame for development;
- expertise available.

In many large-scale projects the issue is not simply whether to use a low-fidelity or medium-fidelity prototyping approach but rather how to plan and integrate the overall development process. A pragmatic or evolutionary approach might be to:

- carry out an initial user requirements analysis;
- produce an initial low-fidelity, paper-based prototype in order to explore the main design concepts and screen layout issues;
- build a medium-fidelity prototype, perhaps a simulation;
- produce a number of vertical, high-fidelity prototypes;
- integrate the vertical prototypes into a complete, fully-functional high fidelity prototype.

Each of these stages is influenced by user feedback and evaluation of the prototype.

In some instances the final prototype may be, effectively, the finished product. In general, however, the prototype is used as one of a number of specification instruments for the software developers, since the overall application may need to be built using a software environment more powerful than that of a quick prototyping tool.

In summary, prototyping can be used by designers and users who are well informed about its benefits and limitations. The process must not be haphazard but must be planned, with explicit version and documentation controls. Most importantly, the organisational climate must be supportive to the concepts of user involvement and cycles of iterative design and evaluation (see Alavi, 1984).

Quick test

1. How is low-fidelity prototyping carried out?
2. What are the main advantages and limitations of medium-fidelity prototyping?

Section 6

End of chapter assessment

Questions

1. The participative design (PD) tradition emphasises the centrality of end-users within the user interface design process. Describe any one PD technique that is based on significant end-user involvement.
2. Discuss both the advantages and problems of involving users in the user interface design process, drawing upon evidence from the academic literature.
3. Describe and discuss the nature of prototyping user interfaces.

Approach to answers

1. There are several possibilities here, such as contextual inquiry, various prototyping approaches, the user-involvement method (Axtell *et al.*, 1997) etc. Marks will be given on the bases of the accuracy and completeness of the description.
2. Advantages: ensures user-commitment to the finished product; increased job satisfaction, sense of ownership of design on the part of users; evidence that user involvement results in better (both more usable and more appropriately functional) products (Keil and Carmel, 1995).

Problems: can stifle designer creativity, lengthen project time; users need training (Axtell *et al.*, 1997). Some users can become 'hostages' to the design team (Damodaran, 1996); selecting appropriate users, rather than surrogates and intermediaries.

3. Identifying user requirements, using appropriate techniques, such as focus groups; developing the prototype with the aid of relevant guidelines; using and evaluating the prototype; and revising the prototype, drawing on the results of evaluation studies. The discussion should emphasise the iterative nature of prototyping and point out that there exist several 'species' of prototypes, such as low- and high-fidelity.

Section 7

Further reading and research

Further reading

Books and papers

- Beyer, H. and Holzblatt, K. (1998) – Chapter 3 'Principles of contextual enquiry'.
 Rettig (1994).
 Dix, A. *et al.* (1998) – Section 5.5 'Iterative design and prototyping'.
 Madsen (1999).
 Norman, D. A. (1998b) – Chapter 9 'Human-centered development'.
 Torres, R. J. (2002) – Chapter 4 'A user-centred product team'.
 Vredenburg (1999).

Websites

InContext. <http://www.incent.com/cd/cdhow.html> InContext is a US-based usability consultancy founded by Karen Holtzblatt and Hugh Beyer. This link gives an interesting description of the processes involved in contextual design.

Prototyping for Design and Evaluation. <http://pages.cpsc.ucalgary.ca/~saul/681/1998/prototyping/survey.html> A comprehensive review of prototyping techniques written by Professor Saul Greenberg, University of Calgary.

SERCO. <http://www.usability.serco.com/services/services.htm> An overview of the services offered by SERCO, a large UK usability consultancy; it outlines the processes involved in user-centred system design.

Further research

1. Plan an exercise to create a paper prototype of a PDA (personal digital assistant). You will need to consider the functionality that will be modelled, the materials you will use and the time scales. It will be necessary to consider the extent of user-involvement in the process, for example the number of users, how representative they are and what activities they are required to undertake.
2. You will probably have to undertake further research in order to produce your plan effectively. When you are satisfied with your plan, carry out the prototyping activity. When the exercise is complete, review the effectiveness of the process with regard to (in particular) the efficacy of the paper-based materials you have used and the advantages and problems associated with involving the users.