Exhibit 1017 – Part 2

Table 4.7 Advantages and disadvantages of direct-manipulation dialogues

Advantages	Disadvantages
Task analogy	May require complex or large software
Reduced familiarization/ learning	May require high-performance graphics display
Readily retained	May require auxiliary input devices (e.g., mouse)
Encourages exploration	Skilled graphics design required
Visually appealing	
Powerful	
Concise	
Design tools available	

graphics packages. Nevertheless, a wide variety of command language, driven software tools were produced in the past for use on minicomputers with character-orientated displays.

Computer-aided design

Similarly, first-generation CAD systems were based upon textual input of commands, for example, circuit analysis programs such as SPICE and HILO require component values to be specified in relation to numbered or labelled circuit nodes. Current systems, by contrast, attempt to provide a more direct representation of the simulation, for example direct schematic capture in the case of circuit analysis, or 2- and 3-dimensional models for finite-element modelling.

4.10.4 Advantages and disadvantages

Table 4.7 shows the advantages and disadvantages of direct-manipulation dialogues. The primary advantage is the direct *analogy* between the task and the system, which gives the user a sense of understanding and control from the start, and encourages *exploration.* In addition, it reduces the time required for the user to become *familiar* with the system, since many aspects should map directly onto his or her existing knowledge of the task, and *learning* is reduced since knowledge of the task is also knowledge of the system, and no independent set of commands is needed. With careful graphic design, direct manipulation can be *visually appealing* and satisfying, and can also be made powerful and concise, thus appealing specifically to expert users. (For example, a double click of the mouse button on a Macintosh application icon is equivalent to typing the command name in a conventional operating system.)

The major disadvantages of direct-manipulation dialogues are that they may require *complex and expensive hardware* to support a suitable graphics display, and *substantial software development* to implement the chosen metaphor realistically.

A variety of *design tools* are available for prototyping graphical direct-manipulation interfaces. Applications such as MacPaint and its equivalent on other workstations have been successfully used for simulating static graphics displays, while Hypercard and derivatives provide some capability to represent dynamic interaction as well. Most windowing systems provide development tools allowing rapid development of pop-up and/or pull-down menus, dialogue boxes, and icon and fount editors (see Section 10.8). The major disadvantages of all these tools is the restriction they place upon the designer, who must conform to the particular 'style' embodied in the tool (e.g., the card metaphor in Hypercard, the specific window and menu format and actions for a window system). The inclusion of first-class graphic design skills in the design team is also an essential requirement for direct-manipulation interfaces: this presents particular problems since the graphic designer's ideas may have to be mediated by a software engineer in order to produce a physical representation if adequate design tools are not available.

4.10.5 Conclusions

The technological disadvantages of direct-manipulation dialogues which have inhibited their development in the past are gradually disappearing with the availability of high-performance, cheap microprocessors and improvements in display technology. At the same time there is an increasing need for larger numbers of users to become familiar with more and more different computer systems. As a result there is a widespread trend towards direct-manipulation interfaces as representing an effective way to provide a system which is appealing to novices and experts alike, and which is easy to learn, encourages experimentation, and is readily retained once learnt.

However, direct manipulation dialogues are not a panacea; in many applications their technical complexity is not justified or may be impracticable or the target user population may be better served by some other dialogue style. Even where direct manipulation is appropriate, other dialogue modes may also need to be introduced to achieve functions which cannot readily be represented metaphorically.

4.11 Design principles and guidelines

Sections 4.6–4.10 have presented specific features of the five major different dialogue styles. In this section, more general design issues are considered which are relevant to all dialogue styles.

4.11.1 Design sequence

Dialogue design, like any other aspect of system design, should be carried out in a top-down fashion. Hebditch (1979) specifies the process of stepwise refinement as follows:

1 Choose dialogue style

The basic dialogue style must be selected from among the five styles discussed previously. The choice between these styles will be affected by the characteristics of the user population (expert or naive users, etc.), the type of dialogue required, and by the constraints of the available technology and application area. This will result in an individual style or combination of styles being chosen for the dialogue.

2 Design dialogue structure

The second stage is to undertake task analysis and determine the user's model of the task on which the dialogue structure should be based. Having proposed a dialogue structure, every effort should be made to obtain user feedback by means of informal discussion or more formal simulation of the interface.

3 Design message formats

Once a satisfactory dialogue structure has been achieved, more detailed attention must be paid to the display layout and textual content. Similarly, detailed user input requirements should be considered with the objective of maximizing efficiency, for example by avoiding unnecessary keying. These issues are considered in the following sections.

4 Design error handling

Having designed how the system will work when sensible data is input, consideration must be given to the ways in which user errors can be made. These include: *input data validation*, where checks must be made that sensible responses or values are given by the user; *user protection*, where the user must be protected from the consequences of his or her own errors (e.g., deleting important files); *error recovery*, that is, the provision of mechanisms for backtracking, undoing or retrying commands which were executed in error; and *meaningful error messages*. Again, these issues are considered in more detail below.

5 Design data structures

Once all aspects of the user interface have been specified, consideration can be given to the internal structure of the system, such as choice of data structures to support the required functionality. These structures should map directly onto the user's model of the system, though their complexity may vary considerably between different applications. For example, the graphics data structures required to support a WIMP interface are quite complex, whereas those required to support a text editor might be much simpler. In either case however, the structures should be derived from the interface specification (which in the case of the editor is, in effect, the user manual), so as to avoid conceptual mismatches between the user's and system's models of the system.

4.11.2 Screen design—text

Stewart (1976) has proposed six major factors which contribute to high quality textual screen layout: these are summarized below.

1 Logical sequencing

Information should be presented to the user in a sequence which logically reflects the user's task, even if this conflicts with the optimum system presentation sequence. If this is not possible, then the rationale for using an alternative sequence should be made explicit to the user.

2 Spaciousness

Clutter on a display greatly increases visual search time: the use of spacing and blanks is important in structuring the display and emphasizing those aspects to which the user's attention should be directed.

3 Grouping

Related items of data should be grouped together to provide higher-level structure to the screen as a whole. This reinforces the concept of 'chunking' and speeds up search times by allowing related items to be treated as a group. Even on character-based displays, auxiliary characters such as line segments may be able to be exploited to emphasize grouping.

4 Relevance

There is a natural desire on the part of the designer to include all which may be relevant to a display. However, displaying the maximum

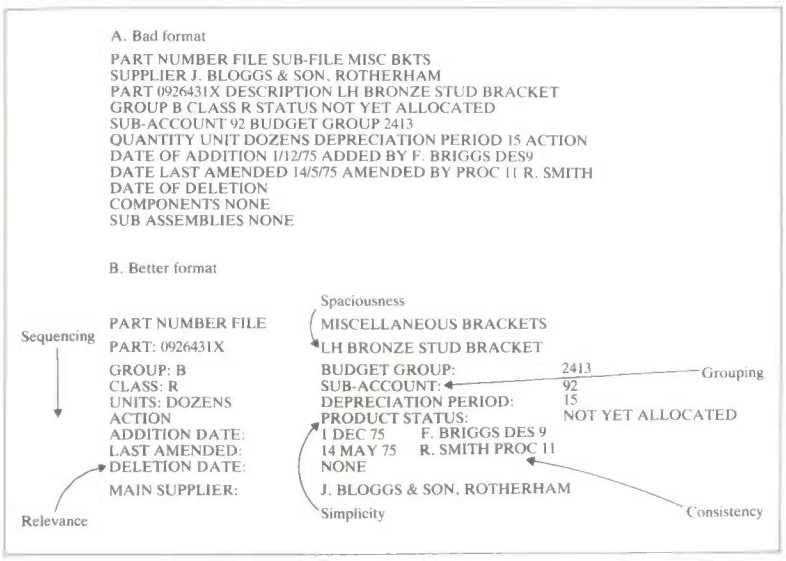

Figure 4.12 Screen textual layout guidelines (reproduced from Stewart, 1976)

amount on the screen is not the same as maximizing the information transfer rate, and it is this latter principle which is of primary importance.

5 Consistency

In frame-based systems where the user views a number of sequential screens full of information, it is important to be consistent in the use of display space, so that the user learns where different types of information are to be found.

6 Simplicity

The overriding consideration in screen design should be to present the appropriate quantity and level of information in the simplest way possible.

Figure 4.12 illustrates these points, and emphasizes the importance of aesthetic appeal in the design of the user interface: the interface is the 'shop window' on the product, and as such is the major factor influencing the user for or against the product.

4.11.3 Screen design—graphics

Graphic design is a well-developed and established area in printing and publishing, but its importance in human–computer interface design is only now becoming established as computer systems are more widely used, and more flexible and controllable display formats become feasible. To a significant extent, screen design is constrained by technological considerations: the response time, display rate, display bandwidth and display type (character- or graphics-oriented) of an interface all impose restrictions on the style of interaction which can be accommodated. For example, the display style will inevitably be much more restricted on a monochrome alphanumeric terminal than on a colour graphics workstation.

Verplank (1988) cites the following five principles of graphical user-interface design, which are primarily derived from experience in the design of the Xerox Star user interface and its predecessors:

1 The illusion of manipulable objects

Effective graphic design involves three distinct components. First, a set of objects appropriate to the intended application must be invented. This involves (hopefully) identifying generic stereotypes of the required objects on which icon design can be based. Next, the skills of graphic design must be used to represent these objects in a convincing way. Finally, consistent graphic mechanisms must be provided for indicating actions on the object such as selection.

2 Visual order and user focus

Graphical interfaces provide the opportunity to exploit very powerful visual stimuli, and features such as flashing, inverse video, strong colours and animation can lead to an overpowering and visually cluttered interface if they are used to excess. One important point is to ensure that the user can readily identify the part of the screen on which attention should be focused: hence, the most powerful attention-getting devices should be used sparingly for this function (e.g., use of flashing to identify the cursor, or use of inverse video on an icon or window title to indicate that it is selected).

3 Revealed structure

Parallels are commonly drawn between direct manipulation and the WYSIWYG concept, in that both are intended to minimize the difference between the observed screen and its effect. A particular problem of this approach however is that it fails to convey the underlying structure of the activity. For example, in textual documents,

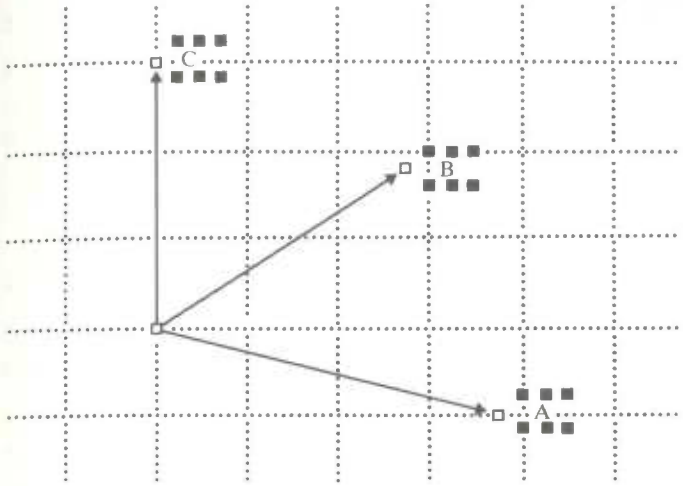

Figure 4.13 Revealed structure of Figure 4.4 showing handles and alignment grid

the layout is typically controlled by non-displayed control characters: although these are irrelevant on the final printout, they may be very important in defining the effect or scope of an action during editing. Typically, this need is dealt with by providing a display mode in which the structure is revealed (for example by showing rulers, e.g. MacWrite or control codes, e.g. Word). Similarly, graphical editors such as MacDraw require a mechanism for explicitly showing the diagram structure, and providing methods of selecting and manipulating elements of it (Figure 4.13). These 'handles' and other hidden elements such as alignment grids form a key element in the design of the graphical user interface, since they convey important information to the user concerning the structure and dynamic manipulation of the screen which is essential for effective interactive use.

4 Consistent and appropriate graphic vocabulary

As with text, it is important for graphic symbology to be used consistently throughout an interface design. There is evidence (Mayes *et al.*, 1988) that interaction with a computer involves a process of 'information flow' where local display information is picked up, used and discarded as necessary to meet functional needs: although the detail of the symbology may only be identified at a subconscious level, inconsistency will inhibit interaction and familiarization with the interface. Figure 4.14 shows some of the graphics conventions used in a MacWrite dialogue box (activation buttons; radio buttons (mutually exclusive set); check boxes (inclusive set); text entry windows). Note

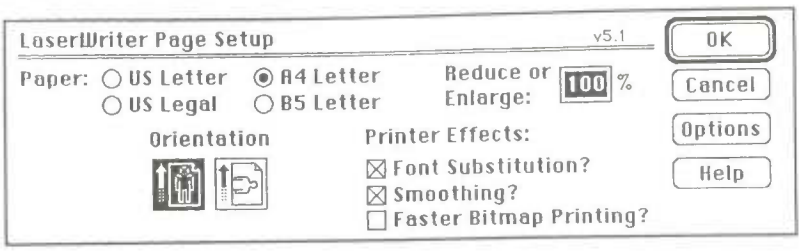

Figure 4.14 Example Macintosh dialogue box illustrating standard graphic symbology

that many Macintosh users readily use these features, which are a consistent part of the Macintosh interface style, without consciously being aware of their underlying definition.

5 A match with the medium

The specific characteristics of the display medium substantially influence the aesthetic appeal of different graphic constructs, and considerable time is required for graphic designers to become proficient in exploiting the capabilities of any particular display style. In the past, screen designers worked within the constraints of alphanumeric character-based displays; currently, many systems are based upon bit-mapped (1 bit per pixel) raster graphics displays; in the future, it is probable that much greater use will be made of grey-scale and colour graphics displays. The experience gained in current bit-mapped displays does not necessarily translate directly into appealing aesthetic design for colour or grey-scale displays, and we can therefore expect a delay of several years before the full potential of these displays is realized.

4.11.4 Response time

It is clear that slow response from a system can have an adverse impact upon the user interface, but the exact response speed required for satisfactory interaction depends to a large extent on the nature of the interaction taking place. Furthermore, *variability* in system response speed also appears to disrupt user performance. Martin (1973) suggests a broad division of response times into 5 categories, derived primarily from Miller's (1968) analysis of 17 situations in which maximum acceptable response times varied widely:

1 > 15 seconds

Response times of this order generally rule out interactive use of the system. The user's attention is likely to be diverted to other activities and only return to the screen on completion of these.

2 >4 seconds

Response delays of this order are poor for short-term memory retention, and thus should be avoided in the middle of a sequence of related operations. They may be acceptable on completion of a major cognitive process when intermediate short-term data can be discarded ('closure'), for example dispatching a job to the computer.

3 >2 seconds

Response delays of more than 2 seconds during interactive dialogues requiring a high level of concentration can be surprisingly disruptive.

4 <2 seconds

A response time of this order is generally considered acceptable for interactive work, for command input, menu selection, form-filling, etc.

5 Almost instantaneous

Almost instantaneous response is required for very tightly coupled activities between the user and system, such as character-by-character response to keyboard input or tracking a mouse or cursor movement on a screen.

4.11.5 Error handling

It is human nature for the designer of a system to concentrate most of his design effort on the way he intends the system to work, rather than recovery from user errors. However, any system will inevitably be used at some time by inexpert users, and thus user input errors will occur and must be handled effectively by the system. Error handling can be subdivided into several distinct requirements, as follows:

System and user protection

Primary requirements are to protect the system from the user, and the user from the consequences of other users' actions (in a multiuser system). In multitasking systems these requirements are generally supported by hardware memory-managements systems and privileged modes. In simpler single-user systems there may be less protection and considerable effort must be devoted to testing all conceivable user interaction to avoid 'crashes': the 'infinite-number-of-monkeys' test. A secondary need is to protect the user from the system: this involves engineering the user interface so that irreversible actions (such as

deleting a file) are not committed accidentally or without careful thought.

Pseudo-errors

Many systems are unreasonably pedantic about the syntax of user input. Programmers' experience of rigid syntax in programming languages gives them an insight into dialogue structures which is denied to other users. Such users will judge the dialogue on the basis of what is 'reasonable' (i.e., readily comprehensible and not ambiguous) for another human, rather than on the basis of some rigid and arbitrary syntax required by the system. For example, the dates

4/1/53 04/01/53 04/01/1953 4.1.53 ...

are all readily interpreted as 4 January 1953 (unless you are American, in which case they represent 1 April 1953!), yet many computer systems are much more restrictive in the date format they will accept. These *pseudo-errors* result from lack of foresight on the part of the programmer: very little extra programming effort can make the interface appear much more friendly.

Error messages

Most computer users will have experienced error messages similar to the following:

```
FATAL ERROR - PROGRAM ABORTED
**SYNTAX ERROR**
WHAT?
INVALID DATA
ERROR OE7 IN DEVICE 080
```

Error messages should be *clear, concise, specific, constructive* and *positive*. The above examples achieve only the second of these requirements. *Clarity* and *specificity* are achieved by providing exact information about the conditions under which the error occurred and exactly what the error was, so that the user has some starting point for diagnosing the reasons for the error. *Constructiveness* implies that, wherever possible, the system will suggest ways of recovering from the error, or correcting it. Finally, error messages should adopt a *positive* and conciliatory tone, and not condemn the user's mistake: as in other commercial areas, the consumer is always right.

4.11.6 Documentation

Documentation includes both offline and online material provided to support the dialogue, and would require a separate chapter to do justice

to the subject. In many engineering projects it fails to get the attention it deserves. High-quality documentation of a project at several different levels appropriate to designers, maintenance staff, training staff and users is a key requirement for any interactive system. An interesting recent trend has been the establishment of documentation companies, to whom software houses subcontract the preparation of documentation for their systems. This approach has two particular advantages: first, the documentation is handled by expert writers, rather than software engineers; second, the documenters are independent from the software development team, and are thus much better able to view the product from the user's viewpoint.

Excellent overviews and guidelines on documentation are provided in Bailey (1982), Chapter 19, and in Shneiderman (1987), Chapter 9.

4.12 Case study: *NEWFOR* teletext subtitle generation system

4.12.1 Introduction

The provision of television subtitles for the hearing-impaired via teletext has been growing since the introduction of teletext in the mid-1970s. Subtitle preparation is a complex and time-consuming task which typically required 30–45 hours per captioned programme hour using first-generation teletext origination equipment. The NEWFOR subtitle generation system was developed following research into the display requirements for captions for the hearing-impaired, and analysis of the operational requirements of the caption preparation process. The system is now available commercially, has been sold to a number of broadcasting organizations throughout the world, and is used for all subtitling carried out by ORACLE Teletext for Independent Television in the UK. A full description of the development of the system described below will be found in Downton *et al.* (1985).

4.12.2 Specification of subtitle requirements

Experience gained from foreign-language film captioning is not directly relevant to captioning for the hearing-impaired, since foreign-language viewers can be assumed to have normal literacy skills, and to be able to use their hearing to identify speakers, mood, sound effects and other subsidiary audible information. The objective of the initial phase of this project was therefore to determine the most effective techniques for conveying soundtrack information to the deaf and hearing-impaired.

A wide range of television programmes were therefore subtitled onto videotape and demonstrated at deaf and hard-of-hearing clubs throughout Britain. Viewers were shown a variety of contrasting subtitle

Figure 4.15 A sample teletext subtitle (courtesy ORACLE Teletext Ltd)

display formats and techniques and asked to express preferences and to comment on and discuss the captions. Over a period of time a consensus set of display guidelines were derived (Baker, 1981), and these guidelines provided a specification for the required display formats and styles of the subtitle preparation system. The guidelines specified format and amount of text per subtitle, subtitle positioning, display options (flashing, foreground/background colour, etc.) and text presentation rate. For example, it was found that subtitles should normally be presented in white, double-height mixed-case characters, left-justified within a black box, as shown in Figure 4.15.

4.12.3 Task characteristics

Subtitles are generally prepared in advance of programme transmission and stored on floppy disk. Figure 4.16 shows the block diagram of a workstation used for this task, and Figure 4.17 illustrates the typical task sequence involved in subtitle preparation (derived from observation of work on first-generation teletext preparation systems and interviews with subtitlers).

4.12.4 Dialogue design

The basic design strategy was to share the tasks of subtitle preparation in an optimum way. In captioning, the skills of the human operator lie

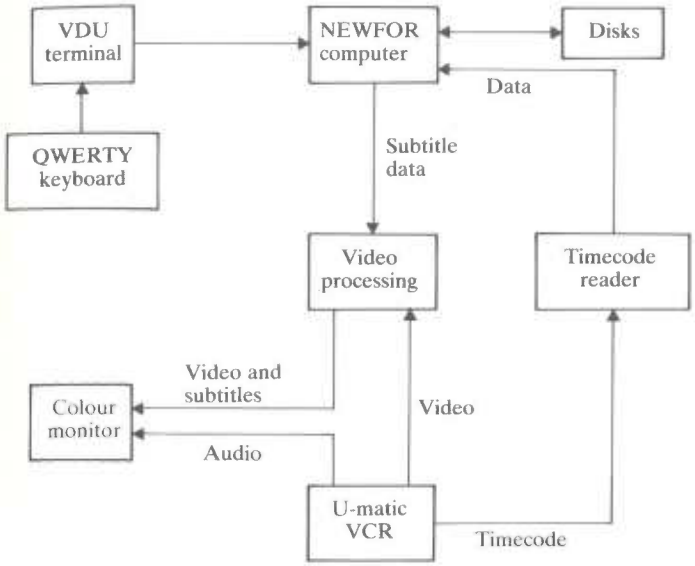

Figure 4.16 Block diagram of NEWFOR subtitling workstation

in linguistic intuition and creativity, understanding of the television programme, and the ability to control the captioning process and assess the results. The captioner may have no detailed knowledge of the technical aspects of teletext. The computer system is suited to handling routine tasks such as text and display format manipulation, data logging and storage and input/output control. Table 4.8 shows the final division of tasks between NEWFOR and the operator: in first-generation systems all of these tasks were explicitly carried out under operator control.

First-generation teletext origination systems mostly used a direct command input mode together with qualifying parameters as a method of control. To reduce the memory and training requirements, a hierarchical menu structure of commands, invoked by typing the first letter of the command, formed the basis of the dialogue with the NEWFOR system. The user's model of the captioning process was reinforced by grouping tasks into operating modes within the hierarchy which corresponded to the preparation strategies adopted by the captioners, as shown in Figure 4.18.

The menu acted as a prompt to the user initially, but as familiarity was gained, particular commands could be remembered by acronyms such as IOT (input offline titles), and the menu could then be bypassed. For novice users, a 'Help' option was available at every command level which gave further details of all commands available at that level.

A standard terminal display style was used throughout all NEWFOR

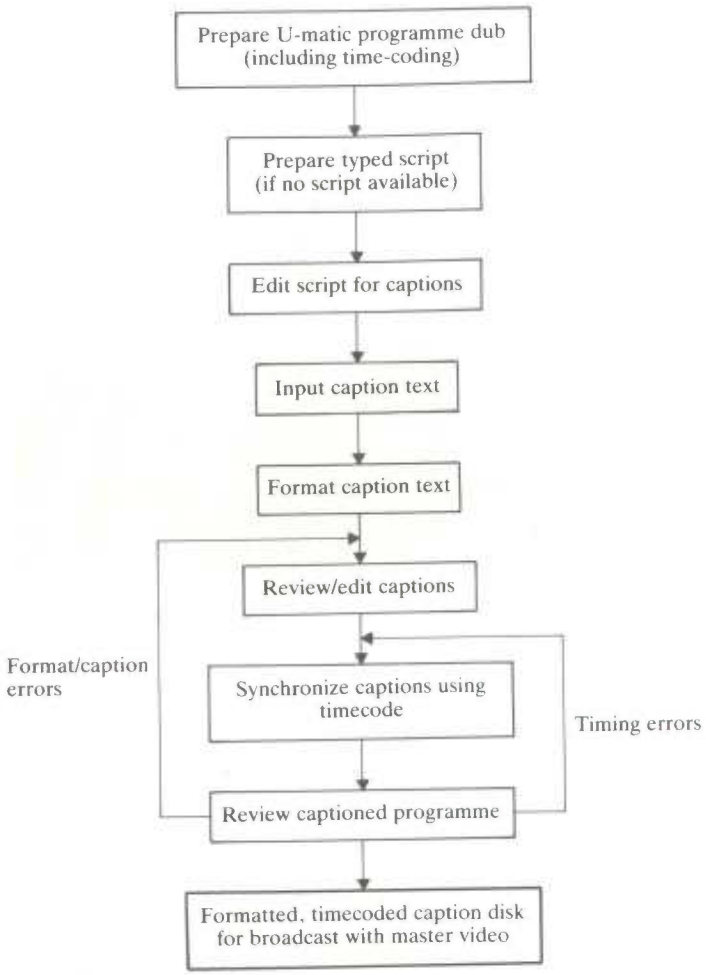

Figure 4.17 Example task sequence in caption preparation

modes, as shown in Figure 4.19. The upper 70 per cent of the screen provided a workspace for displaying current caption input or help information, while the remaining blocked-off display area provided various status displays. Subdivisions within the status block indicate current menu options, operational mode and non-textual characteristics of the current subtitle, such as time code, display time, foreground and background colour, and caption position.

4.12.5 System performance

Performance was assessed using two criteria: training time and productivity. In each case direct comparison could be made with

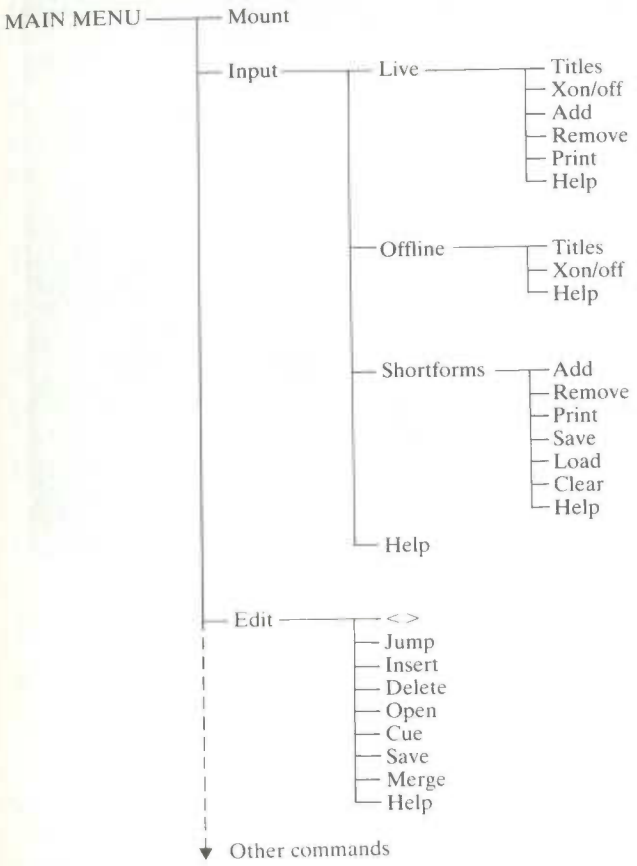

Figure 4.18 Part of hierarchical NEWFOR command menu

Table 4.8 Task division for NEWFOR subtitling system

Captioner	Computer
Text input	Text format (using geometric and linguistic criteria)
Position selection	Positioning of formatted text
Colour selection	Calculation of boxing outline
	Insertion of colour and boxing control characters
Synchronization	Calculation of on-air display time
	Storage of result

Figure 4.19 Sample NEWFOR terminal display

competing first-generation teletext origination systems used for the same process of subtitle preparation.

Training

Initial training to use NEWFOR in the offline input mode was a matter of only a few minutes, since little more than copy typing was required. By comparison, previous systems required an explicit knowledge of teletext display characteristics, including control characters, plus the capability to edit and position subtitle texts. Typically at least one week's training was required to achieve proficiency. A second, more extensive training period was required to achieve full working knowledge of all aspects of the system. This required about one month for NEWFOR, whereas competing systems typically required 2–3 months.

Productivity

The introduction of NEWFOR at ORACLE Teletext Ltd improved productivity from around 25–40 hours per captioned hour of programme material in 1984 to 10–15 hours per captioned hour in 1986. In addition to providing this substantial productivity gain, the offloading of many of the more mundane aspects of caption preparation meant that NEWFOR could also be used for pseudo-live captioning, which would have been quite impossible with first generation systems.

In fact, early versions of NEWFOR were used to subtitle a variety of important live events during its development phase, for example the royal wedding (1981), the papal visit (1982), the state opening of Parliament (1983) and the opening of the Thames Barrier (1984). A modified version of the system linked to a chord keyboard is currently used for live subtitling of the early-evening ITV and Channel 4 News.

References

Bailey, R. W. (1982) *Human Performance Engineering: A Guide for System Designers*, Prentice-Hall, Englewood Cliffs, NJ.

Baker, R. G. (1981) *Guidelines for Subtitling Television Programmes*, IBA/ITCA/Southampton University.

Card, S.K., T. P. Moran and A. Newell (1983) *The Psychology of Human–Computer Interaction*, Lawrence Erlbaum Associates, Hillsdale, NJ.

Daniels, G. S. and E. Churchill (1952) *The 'average man'?* WCRD-TN-53-7, Aero Medical Lab., Wright Air Development Center, Wright-Patterson AFB, Ohio.

Downton, A. C., A. D. Lambourne, R. W. King, R. G. Baker and A. F. Newell (1985) 'Optimal design of broadcast teletext caption preparation systems', *IEEE Transactions on Broadcasting*, **BC-31**, 3, 41–50.

Draper, S. W. (1984) 'The nature of expertise in Unix', in *Human Computer Interaction—INTERACT '84*, B. Shackel (ed.), 465–471.

Gaines, B. R. (1981) 'The technology of interaction—dialogue programming rules', *International Journal of Man–Machine Studies*, **14**, 133–150.

Hebditch, D. (1979) 'Design of dialogues for interactive commercial applications', in *Infotech State of the Art Report: Man/Computer Communication*, **2**, 173–192.

Holmes, J. N. (1988) *Speech Synthesis and Recognition*, Van Nostrand Reinhold, London.

Kidd, A. L. (1982) *Man-machine Dialogue Design*, Research study v.1, No.1, Martlesham Consultancy Services, British Telecom Research Laboratories, Ipswich.

Martin, J. (1973) *Design of Man-Computer Dialogues*, Section II, 87–131, Prentice-Hall, Englewood Cliffs, NJ.

Mayes, J. T., S. W. Draper, A. M. McGregor and K. Oatley (1988) 'Information flow in a user interface: the effect of experience and context on the recall of MacWrite screens', in *People and Computers IV: Proceedings of HCI'88*, 275–289.

Miller, R. B. (1968), 'Response times in man-computer conversational transactions', *Proceedings of the Spring Joint Computer Conference*, **46**, AFIPS Press, Montvale, NJ, 409–421.

Monk, A. (ed.) (1984) *Fundamentals of Human–Computer Interaction*, Academic Press, London.

Moran, T. P. (1981) 'The command language grammar, a representation for the user interface of interactive computer systems', *International Journal of Man–Machine Studies*, **15**, 3–50.

Reisner, P. (1981) 'Formal grammar and human factors design of an interactive graphics system', *IEEE Trans. on Software Engineering*, SE-7, 229–240.

Reisner, P. (1982) 'Further developments towards using formal grammar as a design tool', in *Proceedings of Human Factors in Computer Systems*, ACM, New York, 304–308.

Rosenburg, J. K. and T. P. Moran (1984) 'Generic commands', *Proceedings,*

INTERACT '84, 1st IFIP Conference on Human–Computer Interaction, 245–249.

Shneiderman, B. (1983) 'Direct manipulation: a step beyond programming language', *IEEE Computer*, August 1983, 57–69.

Shneiderman, B. (1987) *Designing the User Interface: Strategies for Effective Human–Computer Interaction*, Addison-Wesley, Reading, Mass.

Stewart, T. F. M. (1976) 'Display and software interface', *Applied Ergonomics*, **7**, 137–147.

Verplank, W. (1988) 'Tutorial notes', *HCI '88 4th Conference of the British Computer Society HCI Specialist Group*, Manchester, 5–9 September 1988.

Weil, A. T. (1965) 'Conversations with a mechanical psychiatrist', *Harvard Review*, **3**, 68–74.

Weizenbaum, J. (1966) 'ELIZA—a computer program for the study of natural language communications between man and machine', *Communications of the ACM*, **36**, January 1966.

5 Knowledge analysis of tasks: task analysis and specification for human–computer systems

PETER JOHNSON and HILARY JOHNSON

5.1 Introduction

Task analysis has emerged as an important aid to early design in human-computer interaction (HCI). It provides an information source from which design decisions can be made, and a basis for evaluating designed systems. Task analysis is an empirical method which can produce a complete and explicit model of tasks in the domain, and of how people carry out those tasks. It focuses design on users' tasks and goals, and the methods for achieving those goals, resulting in improved, more usable system designs.

Although task analysis is the investigation of what people do when they carry out tasks, a method of task analysis concerns more than simply observing how people perform tasks. An approach to task analysis involves a number of aspects:

- a theory of tasks;
- techniques of data collection;
- a method of analysing tasks;
- a representational framework for constructing task models.

In this chapter we describe a theory of task knowledge and then consider techniques of data collection, methods of analysing and generalizing from those data, and a framework for task modelling. The data collection, analysis/generification method, and framework for task modelling put forward are part of our approach to the knowledge

analysis of tasks (KAT). This approach has been developed from earlier work on task analysis for knowledge descriptions (TAKD) (Johnson *et al.*, 1984; Diaper and Johnson, 1989). KAT has been described in Johnson, Johnson and Russell (1988) and is concerned with analysing and modelling the knowledge people possess and utilize in carrying out tasks. It is to be contrasted with task analysis techniques *not* concerned with knowledge, such as ability profiling (Fleishman and Quaintance, 1984), hierarchical task analysis (Annett & Duncan, 1967), and other techniques which have an evaluative role in assessing the complexity of task performance but have no explicit method of task analysis.

The work of Kieras and Polson (1985), Payne and Green (1986) and Card, Moran and Newell (1983) are good examples of current evaluative approaches in HCI which incorporate methods of predicting the difficulty of using an interactive computer system and assume some form of task model. Each of these approaches is capable, in varying degrees, of making recommendations about how proposed system designs can be used in terms of the ease with which users could perform given tasks. There are two important features to these approaches: first, they are not directly concerned with design generation and therefore assume both that decisions about what tasks the system should support have been made elsewhere and that one or more design solutions have already been proposed. Second, they focus on the evaluation and prediction of user performance and do not detail any particular method of task analysis. In contrast, TAKD and particularly KAT identify the knowledge requirements of tasks and are aimed at assisting in the generation of design solutions. KAT may, with further development, also form part of an evaluation methodology.

A similar intention underpins Olson's (1987) approach to a cognitive analysis of people's use of software. However, she does not attempt to identify the knowledge recruited or required by those tasks. Rather, she attaches a view of the cognitive demands of different types of tasks to a form of office system analysis. The cognitive demands considered by Olson are rather simplistic; tasks are described as requiring one or more of the following processes:

- transportation;
- transformation;
- algorithmic decision making;
- judgement-based decision making;
- information correlation;
- information analysis;
- communication;
- information creation.

She assumes that transportation, transformation, simple forms of correlating information, and algorithmic decision making are all suitable for computer support or automation since 'they require actions

that are tedious for the human'. In contrast, judgemental decision making, more complex correlations of information, analysis of information, communication, and creation of information are 'to be allocated to human processing because they capitalize on human strengths.'

One interesting aspect of Olson's approach is her attempt to make explicit some of the criteria by which design decisions can be made from a task analysis which has been undertaken. This attempt is converse to the present and prevalent practice where design is not based on any rigorous task analysis and where decisions about allocation of function are made either from intuition or past, not always successful, design experience. By having explicit criteria (compare the design guidelines contained in Chapter 4 of this book) it is possible to check and evaluate the decisions against those criteria. With only intuition to guide the designer, it is often a matter of 'hoping for the best'.

Rather than leaving design for usability to luck and intuition, we want to identify methods of task analysis which can be used to inform the designer about those factors concerning users' tasks which can influence usability in advance of the designers making inappropriate design decisions. This chapter addresses this issue by describing the method of KAT and its underlying theoretical rationale, and makes recommendations as to how KAT could be used in the design process. The range and complexity of tasks with which we are concerned are not confined to simple keyboard tasks, and are not restricted to physical tasks. We are concerned with tasks as complex and rich as designing the room layout of houses, producing graphs, tables and multimedia documents, producing group documents, running meetings, controlling sophisticated building surveillance equipment, and fashion design. These are just some examples of the complex real-world tasks used as case studies during the development of the KAT methodology.

5.1.1 Theoretical basis for KAT

Before considering the methods for identifying knowledge in tasks, the theoretical underpinnings for three important aspects of KAT are discussed. These relate to the *representation of tasks as concepts, task structure*, and *action and object representativeness and centrality*.

Tasks as concepts

It is assumed that tasks are represented as concepts or general knowledge structures in long-term memory. This is akin to the theoretical position taken by Schank (1982) in assuming that knowledge of frequently occurring events is structured into meaningful units in memory. We have named these conceptual memory structures *task knowledge structures (TKS)*. Empirical support for our assumption can

be found in the work of Galambos (1986). Galambos conducted a series of experiments which show that people recognize and use structures of events, such as the order, the sequence and the importance of activities within the event sequence to understand, explain and make predictions about these events. Further support for our view that task knowledge is represented in memory comes from the work of Graesser and Clark (1985), in which general knowledge structures, goals to causal and enabling states, plans for achieving goals, intermediate states and alternative solutions or paths are all assumed to be represented in a conceptual knowledge structure which is used to interpret events.

A TKS is a summary representation of the different types of knowledge which are recruited and activated in association with task performance. A TKS is related to other TKSs by a number of possible relations; among them *within-* and *between-role* relations. Within-role relations are one form of relation between TKSs in association with a given role. Those tasks which are related because they are performed by the same role will have the within-role relation property associating their respective TKSs. A second form of relation between TKSs is in terms of the similarity between tasks across roles. Each task may be performed differently in one or other respect in the context of a given role. However, a person assuming many roles would have a knowledge structure for each task within a role and also knowledge, not necessarily explicit, of the relations between these tasks across or between different roles.

Task knowledge structures contain *goal-oriented* and *taxonomic* substructures. Goal-oriented substructures represent the goals, plans and procedures for carrying out the task. The taxonomic substructure contains the action–object pairings, their respective properties or features and their role relation links. Further details, and an example of task knowledge structures, are provided later in Section 5.4 on task modelling.

Structure in tasks

Tasks would be unstructured if within a domain all possible components of tasks could co-occur with equal probability combined with all other possible components of tasks. This is obviously not the case; task components or behaviours do not occur independently of one another. Some pairs or even *n*-tuples of task components are quite probable, whereas others are improbable; some groupings of components while being logically possible may never occur in reality. Furthermore, within tasks some task components are naturally carried out together, precede, follow on from, or prime one another. Components of tasks are generally carried out according to some feasible temporal ordering, designated by a plan. For example, a builder who is building a house cannot begin to build until the bricks have arrived. An architect designing the layout of a house cannot design the

upstairs layout until she or he knows how many bedrooms are required. The same architect might have to simultaneously consider certain related task components. For instance, in designing a bathroom layout, the respective position of the bath is considered at the same time as the positions of the wash-hand basin and w.c.

We assume that for the purposes of carrying out tasks a person's knowledge is structured in a similar manner to the structure of tasks reflected in task performance. We represent this structure in terms of task knowledge structures (TKS).

Representativeness and centrality

People's task knowledge includes information about objects, both physical and informational, and their associated actions. Objects and actions differ in how central and representative (or typical) they are. Representativeness (or typicality) refers to how representative an object is to a class of objects. For example, a particular chair may be a good representative instance of 'chairs'. One way to think of representativeness is as a 'good/bad example'. Thus, any particular chair may be a good or bad example of the general class of 'chairs'. Centrality refers to the centralness or importance of the object to the task. This argument is similar to that put forward by Rosch (1985) and colleagues (Rosch *et al.*, 1976) to describe the relations between objects and their categorical representation in memory. Empirical psychological evidence for the centrality of the procedures and action/object representativeness in task behaviour has been obtained by Leddo and Abelson (1986), who found that for tasks such as borrowing a book from a library there were particular task segments which were more central to, and more representative of, going to the library than other segments.

Procedures, subgoals and plans differ in representativeness and centrality to the task by virtue of the typicality or centrality of the actions and objects of which they are composed. For instance, in a similar way to arguing that a 'robin' is a representative instance of the essence of the 'bird' category, so the procedure 'drawing house sketches' might be held to be more representative of an architectural task than 'painting country scenes', since 'drawing' is a more representative action. Both procedures may in some instances be used in the course of achieving the goal of designing a house. In a similar way some procedures may be more central to the task than others. For example, in a tea-making task a vessel in which the water and tea can be combined is considered to be a central object to the task and the action of 'brewing' or 'combining' the tea and water is also central (without it tea cannot be made). However, it does not matter if this vessel is the most typical instance of its class in the task-domain, namely a teapot, or alternatively an atypical instance such as an empty paint can (as has sometimes been used under extreme circumstances). Consequently, a

procedure such as 'brewing the tea in the teapot' is central since it contains a central action and a central object. The distinction between centrality and representativeness is thus that central task elements are considered to be necessary and enable the task goal to be achieved, while representative task elements are the instances from the class of the domain which people most readily or typically associate with a given task.

5.1.2 Identifying knowledge

Having discussed the theoretical assumptions contained in task knowledge structures, it is now possible to consider which aspects of task knowledge should be identified by a task analysis. Different tasks may require particular collections of knowledge, and within a single task a variety of types of knowledge will be required for successful task execution. Therefore, we assume that there are subsets of knowledge which make up a person's total task knowledge.

In identifying the knowledge people utilize in successful task completion, the analyst first needs to identify the person's *goals, subgoals* and *subtasks*; in other words, how the person conceptualizes the *goal structure* of the task. Second, it is necessary to consider the ordering in which the subtasks are carried out: this is determined by the *task plan*. Third, the different *task strategies* (a strategy is a particular set of procedures) must be identified along with the circumstances under which those strategies are employed. Fourth, it is necessary to identify the *procedures* which contain the objects involved in the task and the actions which are associated with them: these are the *action/object groupings*. Finally, the *task objects and task actions* are categorically structured and this structure is a further important aspect of task knowledge which must be identified.

This introductory section has provided a brief summary of the theoretical assumptions from which our approach to task analysis has been derived. The next section describes a methodology for identifying the task knowledge components important for task analysis.

5.2 Knowledge analysis of tasks: KAT methodology (part 1)

This section presents in detail the methods of analysis associated with KAT. There are three parts to the KAT method. First, there are techniques for identifying and collecting data about the knowledge people utilize in performing tasks. Second, there are techniques for identifying the representativeness of a particular task knowledge component and establishing generic task knowledge. This can then be used in the third part of the task analysis method, namely the task

knowledge structure (TKS) modelling process. The methodology does not address the issue of definition of the task domain or how to select sample tasks within a chosen domain. However, in Section 5.7 KAT is related to current practices in system design where the selection of tasks from task domains is considered.

5.2.1 Data collection: applying knowledge-gathering techniques to task analysis

This section is divided into two parts: the first part is concerned with general guidelines for task analysis, and the second provides guidelines for using the various techniques. The next section is concerned with guidelines for identifying task knowledge elements in KAT.

General guidelines for task analysis

Task analysis essentially involves *obtaining different types of information about a task or tasks from different sources using appropriate methods.* Task analysis is an iterative process where the analyst is constantly seeking to identify new information, confirm existing information and reject false information. These general rules of thumb are further qualified by four general guidelines:

1 identify the purpose of the analysis;

2 check the analysis with the task performer(s);

3 analyse more than one person and one task; and

4 make use of more than one technique for gathering knowledge.

Knowledge-gathering techniques

1 *Structured interviews and questionnaires* Interviews and questionnaries are suitable for extracting rules, general principles behind task execution, background information covering low-probability events and the reasons underlying behaviour. Interviews may take less time to carry out than other techniques but they rarely provide detailed knowledge descriptions, and should be supplemented with direct or indirect observation of the task performance of a number of individuals. Interviews are a useful technique for providing an initial view of the task or set of tasks in the domain.

2 *Observational techniques* These are particularly appropriate for providing corroborating evidence and gathering more detailed knowledge, when knowledge is context-bound and when the task involves many individual steps. However, these techniques are

time-consuming, cannot be used in isolation and require inference on the part of the analyst to identify the structure of the task and certain types of objects and actions. *Direct observational techniques*, for example looking over the person's shoulder, are intrusive and may seriously influence the person's behaviour. *Indirect observation*, for example video recording, is less intrusive but requires time and effort in setting up and analysis.

3 *Concurrent and retrospective protocols* Protocols are verbal reports given by the person performing the task: they can be either concurrent with the task performance, or retrospective. Protocols provide detailed information on many aspects of a task, including task goals, task plans, procedures, actions and objects. However, protocols require some inference on the part of the analyst, the responses must be carefully coded and the enterprise is time-consuming. Furthermore, it is not always wise to rely solely on verbal reports since people are not always able to give accurate, precise or reliable verbal reports about their own behaviour. In *concurrent protocols (CPs)* subjects report what they are doing while they are doing it. CPs are appropriate when there is insufficient time to carry out retrospective protocols and when the analyst is interested in what a subject is doing at a given time. It should be noted that CPs may interfere with normal task behaviour in a serious and not always obvious way. In *retrospective protocols (RPs)* the subject is required to generate a durable memory trace while completing the task, and then the contents of the trace are verbally reported after the task has been completed. A retrospective protocol could be given while the task performer observes his or her own task performance, for example, using a video recording. RP reports are appropriate when the analyst requires more reliable information than is available through CP and when the subject can be called back to go over the task recording. Additionally, RPs are appropriate when the analyst is concerned with the reasons for and explanations of any behaviour, cognitive aspects of tasks such as planning knowledge, and the feelings and emotions the person entertains about the task. Both concurrent and retrospective protocols are normally collected along with direct or indirect observations.

4 *Experimental techniques* Below follows a summary of several experimental techniques which may be employed in identifying the similarity of task components, for example the actions and objects, and the features or attributes of those actions and objects. All the techniques described in this section normally require the analyst already to have obtained detailed background information through completed interviews or analysed protocols.

(a) *Kelly's repertory grid* (adapted from Kelly, 1955) The task

analyst must have already identified many or all of the components of knowledge associated with a task or set of tasks. The technique involves, first, selecting a given set of objects (or other task components, e.g., procedures) and then presenting these to the subject in groups of three. The subject is then asked in what way(s) any two of them are alike and different from the third. This grouping and separating process is repeated until all the objects have been presented to the subject. The result is a grouping of similar objects or other components which are assumed to share common attributes. One problem with this technique is that the analyst has to be very careful in choosing which three components are presented at any one time since the contrasting set can have a strong influence on any comparison or grouping. There is also a possibility of forcing a classification outcome which is arbitary, an artefact of the selection procedure, and not representative of the actual relationships between knowledge components in the task domain.

(b) *Card sorting* (adapted from Rosch, 1978) In this technique the analyst is concerned with the similarity of task components. The task components can be objects, actions, procedures, etc. The procedure of this technique is somewhat similar to that of Kelly's repertory grid (above). Task components are entered on cards, one card for each component, and the subject is instructed to group 'similar components', or 'components which are the same kind of thing'. Rosch (1978) and other researchers generally instruct subjects to 'put together the things that go together'. The result of this technique, as with Kelly's repertory grid, is a structuring of similar components which are assumed to share common attributes. Unlike Kelly's approach, card sorting is much less likely to be subjected to experimenter bias.

(c) *Rating scales* Rating scales can be useful in identifying representativeness. For example, the name of each object, or other task component, is entered on a separate card and subjects are instructed to judge the given object for its representativeness and/or centrality to the task on an appropriate scale, for example with the highest number of the scale representing greater representativeness or centrality. An alternative to this procedure is to instruct the subjects to sort the cards into an order of relative representativeness and/or centrality to the task.

(d) *Frequency counts* With frequency counts the analyst must note on how many occasions a task knowledge component is either used or referred to in a task or across tasks. The assumption is that a knowledge component which is more central and/or representative will have a higher frequency score than a component of lesser centrality or representativeness. Frequency counts provide an index which can be used to compare individual

differences across different people performing the same task, and also across tasks. Such comparisons provide some indication of differences in task organization and task plans across individuals. One problem with this technique is that it is likely to be very time-consuming and exacting for the analyst. Furthermore, frequency is only one criterion of centrality, and some task knowledge components may be infrequently used or mentioned but central to task performance in certain contexts.

5 *Other useful techniques* Other techniques which might be used in addition to, or incorporated into, the above are:
   (a) knowledge competitions;
   (b) group discussions;
   (c) multi-choice questions;
   (d) task carried out by the analyst with instruction;
   (e) observation with a knowledgeable person providing the commentary;
   (f) asking for sample outputs',
   (g) cooperating subjects (two or more subjects working in groups).

For further details on these techniques see Welbank (1983).

5.2.2 Identifying knowledge components in KAT

KAT is concerned with identifying a person's task knowledge in terms of actions and objects, and the structure of those objects, procedures, the task plan, task goals and subgoals. The techniques considered in Section 5.2.1 above are now classified according to which aspects of knowledge they elicit most effectively.

Identifying objects and actions

Objects (and their associated actions) used in carrying out the task can be identified from one or more of the following techniques:

1 Selecting objects and the actions associated with them from textbooks, a tutorial session, pilot study or by the analyst herself carrying out the task.

2 Questionning the task performer in a structured interview about the actions and objects, and then listing all the relevant nouns and verbs produced by the person in answering the questions.

3 Asking the task performer to list all the objects they can think of which are involved in the task, and the actions carried out on them.

4 Directly or indirectly observing the person carrying out the task, carefully noting what objects they manipulate and in what ways.

5 Noting all the objects and actions mentioned by the person in either concurrent or retrospective protocols.

entifying planning and procedural knowledge

ıis section summarizes techniques for identifying a person's ıowledge of the task plan, the sequence of carrying out routine ·ocedures, and strategies used in the task.

Asking specific questions in the structured interview This involves asking a person how she or he plans the task, if the same plan is used for any other task, and identifying any modifications required to the plan. It is useful to ask specific questions of the sort, 'What do you do if', for example, 'X goes wrong or fails?' The analyst should also ask whether any particular strategies or procedures exist for carrying out some part of the task, and if so how they are used, and why they are there. A further question to ask is what indicates the end of one part of the task, and what triggers the start of another procedure.

Protocols and observation This involves initially having some knowledge of the task so that the ending of one phase or part of the task and the starting of another can be easily identified. A schema for recording and interpreting the data is required.

Card sorting This technique identifies the sequence of carrying out routine procedures and involves putting known task procedures on individual cards, which the person then sorts into an appropriate order for task execution. The results are then verified with other task performers.

entifying subgoals and subtasks

ıe identification of goals, subgoals and subtasks can be obtained by ıe, a selection of, or all the following four techniques:

Asking specific questions in an interview about what are the goals and subgoals of the task.

Using a textbook, instruction manual, or any other available written material, which decomposes the task into goals and subgoals.

Asking or aiding the person to construct a tree, flow or hierarchical diagram of connected goals and subgoals of the task, making a specific requirement that they label different parts of the task.

Identifying different phases of the task either from observations, concurrent or retrospective protocols. When using observations a phase or part of the task may be identified by pauses. In concurrent or retrospective protocols, it is important to make a note of such statements as, 'Now, I intend/want to ...', etc. The analyst should be sure which referents belong to 'this', 'that', 'it', etc. The task goal

structure can and should be verified by checking it against the goal structure provided by another person.

In this section we have discussed various techniques for collecting task analysis data. Together these techniques form part of the KAT methodology. The next section is concerned with analysis and generalization of the collected data.

5.3 Identifying representative, central and generic properties of tasks: KAT methodology (part 2)

This section is concerned with identifying representative, central and generic properties of tasks within a given domain or across domains. Some task components are more representative/typical of a task than are others. Central task components are those necessary to successful task execution: without these central components the task goal will fail to be achieved.

Generic task components, on the other hand, are those common across a number of task performers. The term 'generic' in the context of KAT relates to general rather than specific elements of tasks identified by the analyst. The essential function of identifying generic task components is to reduce variation both across subjects, across the technology and across instances of similar tasks in the domain(s).

5.3.1 Representativeness and centrality

Identifying representativeness and centrality

Task knowledge components can be structured in terms of their representativeness and centrality to the task, using one or more of the following methods:

1 *Frequency* Count the frequency of times a particular task component is referred to, in either interviews or protocols. The assumption here is that the more representative/typical components will be the most frequently referred to.

2 *Ratings* The analyst may use rating scales where the name of each task knowledge component is presented on a separate card or other medium, and the person asked to judge the relative representativeness/typicality or centrality of each component on a scale of, for example, 1 to 5.

3 *Ordering* Presenting task components on cards as in (2), the subject is required to sort the cards into an order of increasing representativeness or centrality of the task.

4 *Recall* The analyst instructs the person to recall from memory all the task components. The order in which they are listed may reflect the order of centrality of each component within the task. The resulting lists recalled (one from each person) can then be correlated to determine the degree of agreement of task component centrality across the sample population.

5.3.2 A method of generification

A method of generification must be capable of identifying generic actions, objects, plans and procedures. Generification is the process of abstracting from instances of tasks, people and technology and thereby reducing the variance in task performance.

Generic actions and objects

The first step in identifying generic actions and objects is for the analyst to construct two separate lists, one for the actions and one for the objects that have been manipulated, mentioned or referred to in some way by the task performer(s). These lists will contain disparate (and often repetitive) information from each task performer over one or a range of tasks.

For example, in the task analysis of an architectural task, namely 'designing the room layout of houses', lists containing all the actions and objects suggested by two different architects were obtained. In many cases the same actions and objects were manipulated or referred to on both lists. Examples of the objects were *plans, symbols, windows, pipes, appliances, doors, pens, rulers*; examples of the actions were *draw, rehang, check, reposition*, etc.

The second step is to reduce the lists generated above to comprehensive and non-repetitive lists with each action and object appearing once only. However, the original lists are also retained, as they provide a measure of frequency of the respective objects and actions in the task and hence may be of use in the identification of representative actions and objects.

The third step is to choose generic actions and objects and is achieved in one of two ways. The first method is to assume a critical value or threshold of frequency across subjects and tasks. The analyst must decide at what level the frequency threshold is to be set in order to judge if something is or is not generic. (Caution must be taken in setting this level as some or possibly all the objects and actions may already be generic by virtue of being identified.) For example, it may be decided to treat an item as generic if it is referred to by two or more task performers. If this yields an unmanageable (i.e., too large) list of generic actions and objects then the threshold may be raised. Setting the threshold relies to some extent on the analyst's intuition and experience;

however, the analyst can systematically experiment with different threshold values. Threshold setting is an iterative process. Essentially, the essence of the approach is to treat frequency across people and tasks as an indicator of generic terms.

Alternatively, the generic actions and objects can be selected by grouping like terms. The assumption here is that the comprehensive and non-repetitive lists contain all actions and objects involved in the task and that these are then grouped. Grouping all like terms involves the following:

1 The analyst(s) relying on intuition and using an iterative procedure to associate a particular term with other similar terms. Similarity is determined by attempting to re-express the original task description in terms of the alternative or target term. If the alternative term was 'adequate' then the two are said to be similar.

2 Grouping by independent judges. The analyst asks one or more judges to sort objects and actions into groups with the instruction to 'group together the actions (objects) which go together, or are the same kind of action (object)'. The results of each judge's sorting can then be correlated to identify the agreed, generic task components.

After the groupings have been produced, the next step is to identify a generic label or term which might cover all the individual elements in a particular set. These labels then represent the generic task elements.

In the architectural task 'designing the room layout of houses', we used the threshold level method to identify generic actions and objects. This procedure was used since there was a time constraint and generally it is quicker to use a threshold value than to group like terms. By using this method we identified many generic actions (such as 'draw') and objects (such as 'plans'). The 'grouping like terms' procedure has advantages over the threshold method since it provides an opportunity for the task performer to judge whether the generic actions and objects identified are indeed generic. If the threshold method is used then some checking of generic elements can be achieved by involving the task performers in a validation process.

The fourth step is the validation of the generic elements. To validate the generic elements, all the actions and objects are listed separately from the generic labels. The task performers are then instructed to identify to which generic group each action or object belongs. If the action or object is not adequately covered by a generic title then the task performer is free to supply an alternative group title.

Generic procedures, plans and goal structure

Procedures, plans, goals and subgoals are considered together here. One obvious way in which procedures, plans, goals and subgoals differ from

generic actions and generic objects is in terms of the number of alternative choices available to the person. For example, there may be a large number of objects and actions which have to be manipulated in performing a task. However, there are likely to be a smaller number of alternative plans involved in carrying out a task depending on circumstantial constraints and very few alternative task goals and subgoals. Task plans have generic features which are always present in carrying out a particular task, but there will also be specific features which make the plan and the ordering of procedures flexible and which depend on differing circumstances and contexts. The procedure for identifying generic procedures, plans, goals and subgoals is different from that of the identification of generic actions and objects, and consists of the following four stages:

1 List all the components and sequence-related details of plans, procedures, goals and subgoals which result from carrying out the identification procedures in Section 5.2.2 ('identifying planning and procedural knowledge', and 'identifying subgoals and subtasks') above.

2 Verify these details with a number of task performers by asking if the procedures are appropriate and if they are in the correct sequential order, or alternatively by having activities written individually on cards that task performers must sort into an appropriate order for carrying out the task.

3 Include in the generic description all generic procedures, provided by a chosen number of task performers and instances of the same tasks.

4 Verify this generic description with a sample of task performers by asking if this is how the task is usually carried out and by noting under which circumstances exceptions are appropriate.

5.4 Task modelling: KAT methodology (part 3)

5.4.1 Constructing task models

A task model is a model of the user's knowledge of a task. The aim of task analysis is to identify the functional attributes of a person's task knowledge (see Johnson *et al.*, 1988). In this section we provide a methodology to aid the analyst in constructing task models.

We demonstrate the task-modelling method here by constructing a model in the domain of the architectural task referred to previously. Four stages have been identified in the construction of a task model. These are as follows:

- construction of a summary task knowledge structure (TKS);
- construction of goal-oriented substructure;
- task procedures;
- construction of a taxonomic substructure from the generic task actions and objects.

Construction of a summary task knowledge structure (TKS)

In this, the first stage in task modelling, we assume that the analyst has identified the task(s) to be analysed, and then collected the appropriate data using the procedures outlined earlier in Section 5.1. These data should then have been subjected to a generification analysis by which common task knowledge components were identified (see section 5.2). These common task knowledge components then make up a subpart of the TKS.

A TKS includes a summary of all the knowledge a person possesses about a task and gives the task analyst the opportunity to label the identified knowledge. One significant advantage of modelling knowledge in this way is that links to knowledge required by similar tasks can be made. Through making such links commonalities between task knowledge may be identified, either by within-role relations or alternatively by identifying common task elements, for example objects, actions and/or plans. For example, a computer system and its user interface might be required to support several types of tasks both within and between roles. The summary TKS identifies what the common properties of knowledge across a variety of tasks are, and thus what the common requirements of the system might be.

Within a summary TKS there are goal-oriented and taxonomic substructures and procedures. The next section describes the construction and properties of goal-oriented substructures. Figure 5.1 is a summary TKS for the architectural task 'design the room layout of a house'.

Construction of a goal-oriented substructure

Planning activity involves satisfying a set of goals and subgoals by a prespecified sequence of procedures of actions upon objects. Therefore, plans are inherent in goal-oriented substructures. A goal-oriented substructure can be represented by a network of structured goal nodes which direct sequences of events which unfold over time, and eventually satisfy subgoal nodes. Goal nodes can vary in hierarchical level. An assumption made here is that goals and subgoals can be represented by nodes with links between them. Nodes can be treated as conditions, as states or as desired states (subgoals). Subgoals can also be hetirarchically and concurrently related to each other.

The goal-oriented substructure 'calls up' appropriate knowledge from

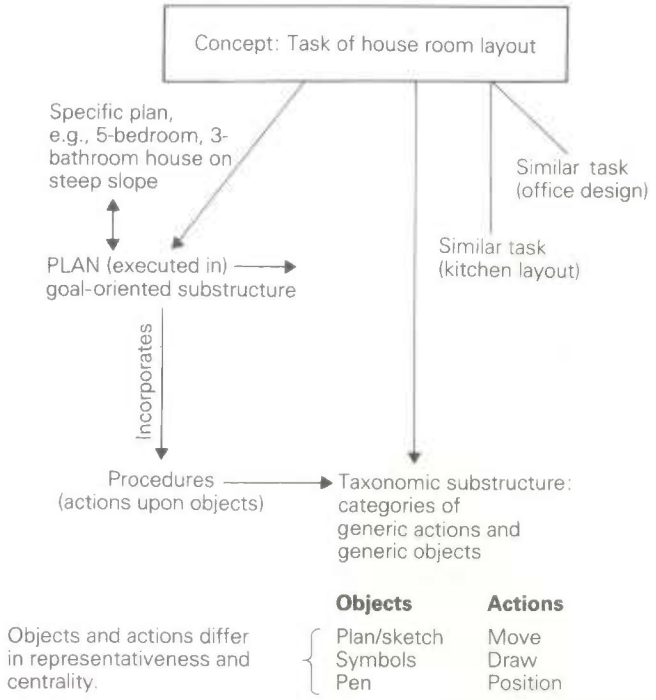

Figure 5.1 Task knowledge structure for the 'design the house room layout' task

the taxonomic substructure by the use of procedures. Associated with subtasks are sets of procedures which have to be executed in order to achieve subgoals directly or indirectly. It should be noted that any subgoal may give rise to further planning activity and subsequent subgoals and thus be indirectly related to a procedure set. Figure 5.2 is a subpart of a goal-oriented substructure for the room layout task.

Task procedures

Task procedures define the ordering of action object combinations in the execution of a given subgoal. The procedure contains sequence, iteration, and other control information which affects the execution of a subgoal. Task procedures are collected together in a procedure set (rather like a macro procedure). Task procedures are executable behaviours. The procedures can be modelled by production rules, by psuedocode or alternatively by frame-based representations as in Johnson *et al.* (1988) and Keane and Johnson (1987). Each task plan ultimately requires an appropriate procedure set before it can be realized in actual behaviour. Task procedures are the process by which the taxonomic substructure is activated.

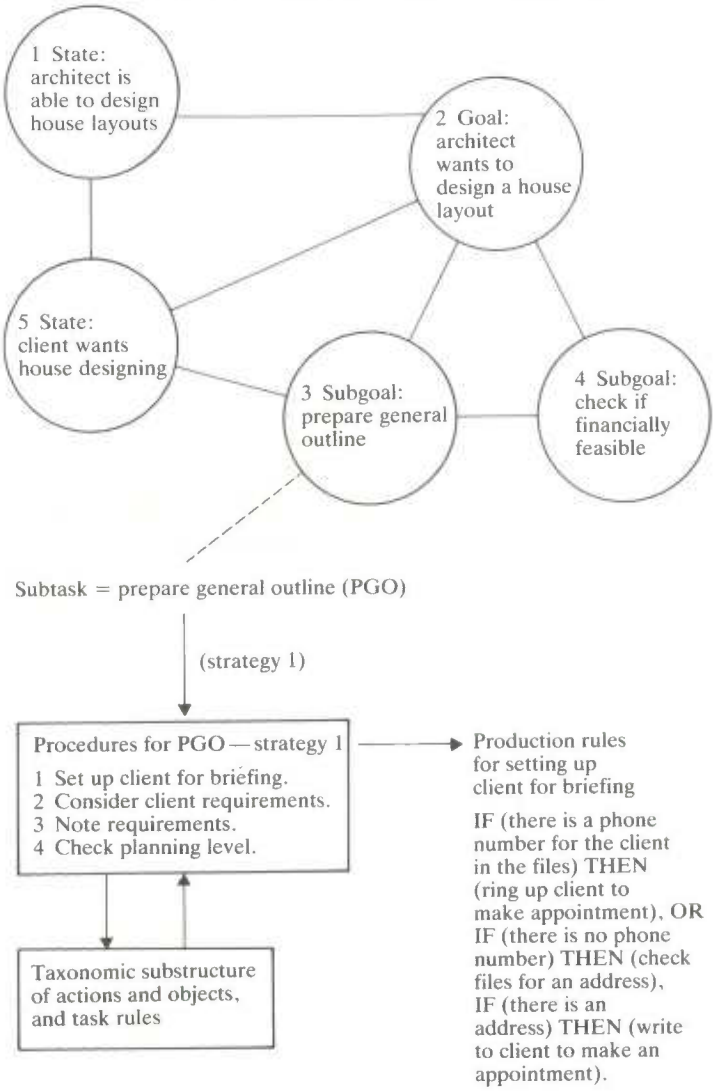

Figure 5:2 Subpart of a goal-oriented substructure of the architectural task

Not only may the task be decomposed in different ways; there may also be a choice between a number of different strategies which are context-dependent competing sets of procedures. One set of procedures may be more appropriate than other sets. Strategy appropriateness will be affected by contextual information and the circumstances under which the task is to be executed. Single procedures in a given strategy may differ in how central they are to the task as a whole. Some procedures will be so central to the task that a failure to execute will result in the task being unsuccessful.

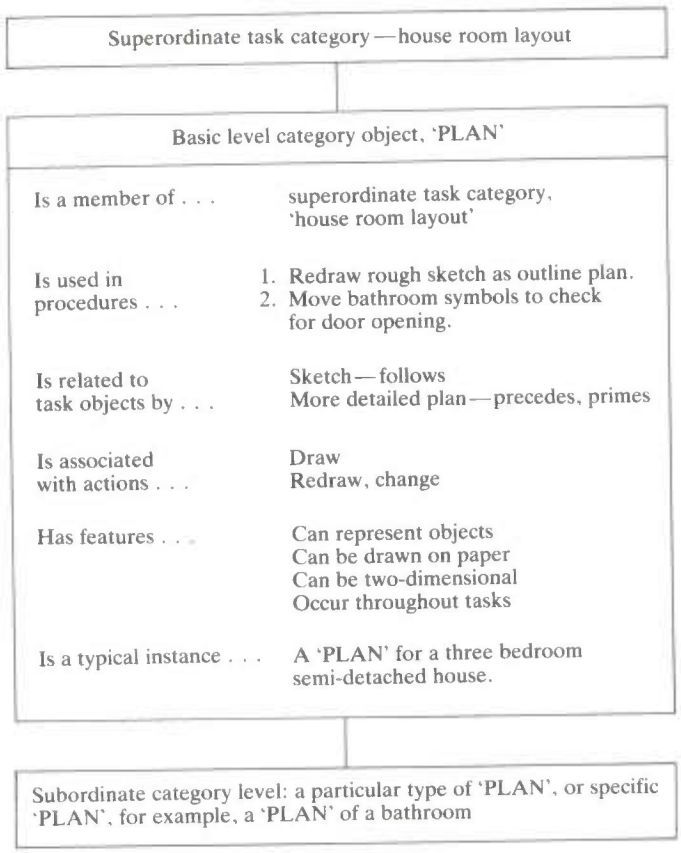

Figure 5.3 Taxonomic substructure for the architectural task illustrating the basic-level object 'PLAN', and its relations to the superordinate category levels

Construction of a taxonomic substructure

The taxonomic substructure contains knowledge about generic actions and objects and the relationships between them. The taxonomic substructure has three levels of abstraction (see Rosch *et al.*, 1976) but is not a static hierarchy.

The top level of the taxonomic substructure is the superordinate task category. In Figure 5.3 the superordinate category for the architectural example is 'house room layout'. The basic level of the taxonomic substructure contains the objects and their associated actions which constitute the superordinate task category. The basic level task category represents knowledge including the following:

- in which task procedures a category member is used;
- which other task objects a category member is related to, and what

that relationship is (i.e., whether the category member primes, precedes, follows or is carried out in conjunction with other task objects);

- which actions are associated with a category member;
- what features or properties a category member possesses;
- the usual circumstances under which a particular category member occurs (for example, whereabouts in the task the category member is manipulated);
- central and typical objects and actions.

Other features of knowledge may also be included at the basic level.

The bottom level of the taxonomic substructure is the subordinate task category which contains a particular type of the object represented at the basic level. The hierarchy is shown in Figure 5.3 using the example of a 'plan' (i.e., sketch plan) as the basic level category object from the architectural domain task.

5.5 Summary features of KAT

The KAT methodology presented here relies on category theory, general knowledge structures and other aspects of cognitive psychology to provide the rationale for making design recommendations and improving design usability. We believe that existing user knowledge will be maximized, leading to quicker learning, potentially fewer errors and easier task execution if the design of the system represents the task components which have been argued as forming a part of a TKS model. If representation of all task components is not possible then the most representative and central actions and objects which have been identified should be represented. A prediction here is that the usability of the system will decrease proportionally to the number of representative and/or central objects or actions not represented to the user at the interface.

Moreover, the user interface design should support the usual sequence for carrying out the task(s) as a default while allowing sequential flexibility by supporting the different, previously identified strategies, which are employed by task performers in usual circumstances of task execution.

5.6 Making design recommendations from KAT and TKSs

The TKS model contains useful information which can be used to influence the design of a computer system. Consider the design of a

computer-based messaging system to support the common task of 'arranging a meeting'. This is a task common to architects, managers, secretaries and other job-roles. The manner in which the TKS model may influence design relies upon the overriding assumption that a computer system will be easier to use if the users are able to transfer some of their existing task knowledge to the newly created environment. This assumption underpins the use of metaphors such as 'desktops' or 'forms', in which the system design attempts to retain some identifiable links with a user's assumed extant knowledge about real desktops and paper-based forms.

However, it is clear that a metaphor is only one mechanism by which transfer of extant knowledge might be facilitated; furthermore, the way in which a metaphor might function is itself the subject of some debate. Moreover, not all aspects of a person's extant knowledge will be relevant or transferable to the new environment. For example, the knowledge of how to dial and use a telephone may have little relevance in supporting communication by a textual computer-based messaging system. Nevertheless, the knowledge a person utilizes in asking questions, making requests, or providing answers would be applicable to both the old and new environments for communication and could (should) be supported in the new environment.

5.6.1 The TKS design support hierarchy

The TKS model identifies the conceptual knowledge structures which a person is assumed to access when carrying out any task. Having constructed a TKS model the analyst has identified a number of important properties of user's task knowledge of benefit to the system designer. At the highest level the TKS shows the relations between tasks and roles. This information provides the designer with a view as to the different kinds of tasks the system will support by virtue of common task/role properties, and also how different roles might expect to have access to the same task functions and to those task functions specific to particular roles. Task role information is also of use to designers who may be concerned with configuring a system to suit the needs of a particular organization, since it shows the task/role match of the organization.

The next level of the TKS represents an overall summary of the plan, the procedures, and the objects and actions people associate with a particular task. This information may be of interest to the designer in so much as it provides an overview of particular contexts in which specific procedures might be used. It could also be used to provide the user with a summary representation of how the designer expects a task to be carried out.

At the next level of representation the TKS model identifies a person's knowledge about the identified goal of the task and the plans