User Modelling for Evaluation of Direct Manipulation Interfaces

Mark Vincent Springett

Submitted for Examination of Doctor of Philosophy

Department of Business Computing,
City University,
London

July 1995

TABLE OF CONTENTS

Thesis Figures and Appendices

Figures

Appendices

Acknowledgements

I would like to thank Professor Alistair Sutcliffe for his dedicated help and assistance as my main supervisor. I would also like to thank Dr Gordon Rugg for his invaluable contribution as my second supervisor. Thanks as well to Logica Cambridge for their funding and support in the first three years, and to EPSERC for their funding, and particularly Mr Ian Clowes for his contribution to the work. I thank my parents, Peter and Brenda Springett for giving me the support necessary to persevere with the thesis through difficult circumstances. I also thank Mrs Liz Bromley, Ms Karen Coates, Mr Paul Collins, Mr Mark Dransfield, Mrs Anne Doubleday, Mr Peter Faraday, Dr A.Simon Grant, Ms Angela Lucas, Dr Neil Maiden, Mr Gareth Martin, Mrs Wendy Martin, Mrs Uma Patel, Mr Rod Rivers, Ms Michele Ryan, Ms Rachel Soper, Mr Graham Stirling, Mrs Irene Stockdale, Mr Steve Woodcock and Dr Dan Wright for various acts of help and support.

Declaration

The author grants the power of discretion to the university library to allow the thesis to be copied in whole or in part without further reference to the author.

Abstract

This thesis applies models of user action to usability evaluation of direct manipulation interfaces. In particular, the utility of a Model of Action for assisting novice evaluators in usability tests is investigated. An initial model of user action is proposed, based on the theory of action proposed by Norman (1986). This model includes a description of knowledge sources used in interaction, error types and user responses to errors. The model is used to interpret data on user behaviour and errors in an empirical study of MacDraw I. This study used the Protocol Analysis technique proposed by Ericsson and Simon (1984). Protocol evidence shows that the search and specification stages of user action could usefully be treated as separate in terms of user knowledge recruitment and the nature of system support. The Model of Action is then expanded and modified to account for the empirical findings. The new model distinguishes knowledge-based, rule-based and skill-based processing in Direct Manipulation (DM) interaction, using the distinction drawn by Rasmussen (1986). These processing levels are explicitly linked to types of presentation technique and categories of user error. This is developed into a technique for determining system causes of usability problems. A set of mental dialogue tokens (roles) are developed to assist novice evaluators in the interpretation of error causes. Roles are linked to types of user error in the cycle of action in a diagnostic model. This model forms the basis of a budget method for use by novice evaluators, named Model Mismatch Analysis (MMA). These developments are tested by a two-tier study of user performance on Microsoft Word. The empirical evidence validated the taxonomy of errors, and tests the utility of five retrospective data analysis techniques. A study of novice evaluator performance is reported, comparing the MMA method to the Usability Checklist proposed by Ravden and Johnson (1989). The MMA method is shown to be the more efficient approach. To summarise, models of Direct Manipulation action are shown to assist novice evaluators both in the diagnosis of usability problems, and the selection of remedies.

Chapter 1- Overview

1.1. Introduction

The thesis brings together two related research aims. One is to give Direct Manipulation design a firmer theoretical footing. This is approached by developing a model of the user, and of interaction. Related to this is the development of methods suitable for use by designers in industry. This is approached through the development of a practical method of evaluation which can be used by members of the design community. The project aims to package user modelling knowledge in a form that supports swift evaluation by novice evaluators.

1.2. Direct Manipulation Interfaces

Direct Manipulation (DM) is a style of user-computer interaction in sharp contrast to early user interfaces. It was the early interface styles (e.g. command languages) that became the focus for Human-Computer Interaction (HCI) research. Early models of computer use were targeted towards this type of system (e.g Card et al 1983, Payne and Green 1986). In the late eighties the need for research to address other styles of interface became apparent (Suchman 1987, Mayes et al 1988, Norman 1986, Shneiderman 1987, Laurel 1988). In particular, the influence of display as cues for interaction became a central theme. Models of users and of interaction were developed to account for this (Norman 1986, Lewis 1988, Howes and Payne 1990, Tauber 1990).

The potential of DM interfaces is demonstrated by the success of Apple and Microsoft Windows products. DM interfaces have characteristics which reflect a range of modern interaction techniques. These include mouse input, windows, pop-up and pull down menus, and the use of icons. DM has the added characteristic of direct action on interface objects. This represents a fundamental change in the nature of user action (Draper 1986, Hutchins et al 1986). The user's intentions are expressed directly on objects rather than being mediated through an interface language. This affords the possibility of creating highly interactive empowering technology for a large range of user types. A number of difficult applications are made quick and easy using DM systems. Whilst this style of interaction is less powerful in certain applications than command-line systems such as Unix (e.g. text processing) it can make technology extremely accessible. DM style interaction can broaden the use of sophisticated and powerful technology, bringing considerable commercial benefits to organisations.

Whilst display-based interfaces and interaction has been the subject of focused attention in recent literature (e.g. Payne 1991, Nielsen 1990), there are no detailed models describing the dynamics of DM interaction. Further investigation of DM interaction and user behaviour is therefore required.

1.3. Models in Design

The design of an interactive system necessarily embodies a model of the user (Carroll 1991). DM systems rely heavily on communicating concepts to the user by matching representations to the user's model of concepts (Laurel 1988). Therefore, user modelling is a key element of designing and evaluating DM systems. A number of user models exist in the literature, but there are two shortfalls. The first is a marked absence of meaningful links between formal models of users (and interaction) and methods of design. Modelling typically takes aspects of users and makes precise definitions (e.g. Payne and Green 1986). By contrast, design guidelines (e.g. Smith and Mosier 1986) tend to be general rules of thumb. In addition, the user modelling literature has yet to account for the specific nature of DM style interaction. Some high-level accounts exist (Hutchins et al 1986, Shneiderman 1987), but these accounts possess insufficient detail to effectively inform design.

1.4. Evaluation

Evaluation may be applied at various stages in the design process (Sutcliffe and Springett 1992). Evaluation techniques may be used to test completed products (summative evaluation). They may also be applied to early design artefacts, design mock-ups or prototypes (formative evaluation) (e.g. Shackel 1986). Early user models (e.g. Card et al 1983) were designed to make formal evaluations of interfaces. More recent attempts have tried to apply user models to practical evaluation techniques (Lewis et al 1990). However, these models could not give complete accounts of interaction, or isolate usability problems. Field-orientated studies provide a contrasting approach (e.g. Whiteside et al 1988). However, these provide data that requires processing and interpretation. Also, this type of study may fail to capture important details, even when a large volume of data is collected.

There is no current method which integrates practical analysis of real users with user modelling knowledge. This is despite the fact that practical and theoretical research (Suchman 1987, Mayes et al 1988) suggests that 'modelling the user' must account for

how users respond to the display. In other words, an accurate model of action should be an 'interaction' model, describing current and previous influences on user behaviour. Therefore, user response to the display must either be accurately predicted, or elicited during interaction.

1.5. Evaluation in Industry

Evaluation is acknowledged as an important part of the design process. Despite this, industry's response to the need for evaluation is patchy. Some reports have suggested that industry is reluctant to use evaluation techniques in design (Bellotti 1988). This is reported as being partly due to the expertise required to use some methods, and partly because of the amount of time and resources required to perform evaluation (Rosson et al 1988). A number of contemporary methods require considerable expertise and effort to apply (e.g. Card et al (1983), Keiras and Polson 1985). This renders them largely inaccessible to the design community. Another problem is that industry tends rely on field reports to test products, and formation of new product versions. The viability of in-house evaluation may only be accepted if a tangible improvement to current practice is offered.

The typical background of members of the design community does not include relevant evaluation expertise. Higher education in Computer Science and related disciplines may not include any HCI training. Recently, HCI modules have been integrated with Computer Science in some universities. However, the background knowledge required is unlike other branches of the discipline. Only some of the larger organisations (e.g. British Telecom, IBM) have Human Factors professionals on their payroll. Consequently, there is a need for methods in industry which can be used by people who do not have HCI expertise.

At present there are no methods which both employ cognitive science models, and are usable in the absence of an HCI/cognitive science expert. There are moves towards such a development from the academic community (e.g. walkthrough methods (Lewis et al 1990)), but this is still far from complete. User Modelling knowledge requires a method that can bridge the gulf in expertise. Further investigation is required into the content and format of such a method.

1.6. Overview of the Thesis Approach

The thesis investigates current contributions in the literature to the understanding of display-based interaction, and approaches to modelling users. The less structured but more paradigm specific literature on DM interaction is also investigated for leads about critical and distinguishing factors in DM use. This is augmented by study of cognitive science issues relevant to the usability of DM systems. These include models of environmentally-based action, display-based learning, the use of metaphor and analogy, and task-based memory.

Contemporary approaches to evaluation are investigated in order to establish an appropriate format, both for conducting investigation within the thesis, and developing an appropriate evaluation method. An approach employing Norman's (1986) theory of action for interpretation of user data is described. This theory is developed into a model of DM action, incorporating accounts of user error responses, and the type of errors associated with the action cycle. The model is investigated empirically to validate its overall claims about the nature of DM action, and to further elucidate and refine its description of DM action.

The investigation of the model employs user-based studies of novice users performing a task. Protocol Analysis (Ericsson and Simon 1984) is used to gather data. Also, the suitability of the protocol analysis method for model-based evaluation is investigated. A method for analysing mismatches between the user's model of the task and the system model is introduced, based on the work of Keiras and Polson (1985).

Evidence from the empirical studies is used to expand the description of DM. The description of mental processing levels described by Rasmussen (1993) is used to distinguish between use of known procedures, learning from examples, and guesswork in novice interaction. Error types associated with the cycle of action are refined and applied to points in the cycle best representing the root cause of the problem.

The model is developed into a prescriptive version, in which design problems can be traced from user errors. The sequence of action is described in terms of abstract interaction phases (activities and roles). These describe user needs at specific points in the cycle of action. The phases are linked to types of design problem. This model is linked to types of error phenotype (Hollnagel 1993). The phenotypes can be identified and traced to causes using visual and verbal evidence from user studies. The model is integrated with the York Manual technique (Monk et al 1991, Wright and Monk 1991), itself a variant of Protocol Analysis (Ericsson and Simon 1984).

The method as a whole is tested for effectiveness, efficiency and usability in a comparative, user-based study. Ten subjects acted as observers, monitoring novice users performing a task on the Word 5.1. package. Ten subjects acted as evaluator using an alternative, questionnaire-based method (Ravden and Johnson 1989). The subjects were selected on the basis of having similar background and knowledge to the target method users in industry. Performance of evaluators using the methods was compared. The studies suggested that the approach is potentially useful in industry. It compared favourably with the other tested method for effectiveness, efficiency and usability. The study also raised some more general points about the format of evaluation methods. The issues raised contribute to contemporary debate on the usability of evaluation methods.

Chapter 2 - Review of User Modelling and Evaluation Work

2.1. Introduction

This chapter discusses the nuances of DM-style interaction, and the implications for user-model and evaluation method development. The contrast between DM and more traditional interfaces is described, with particular reference to the role of the display as a determinent of user behaviour. The suitability of contemporary user models in this context is then discussed. Similarly, contemporary evaluation approache are assessed for their likely effectiveness and coverage when applied to DM interfaces. These include methods derived from the established user models, and more recent methods which are designed to address DM issues.

2.2. The Direct Manipulation Metaphor

2.2.1. Introduction

Direct Manipulation is a style of interaction that departs fundamentally from traditional interaction design (Shneiderman 1982, 1983). Most theoretical HCI developments had been developed for, and through study of, command-style interaction, or heavily constrained menu-style interaction. Therefore, the contrast between DM and other interaction styles is of central interest. Contemporary user modelling and evaluation approaches are reviewed with reference to this contrast.

2.2.2. The Qualitative Contrast

Shneiderman (1986) reports seven qualitative aspects of interaction on DM style systems. All seven are reported as positive aspects of using the system, particularly for users who would not be considered computer experts. These include mastery of the system, confidence in their capacity to retain mastery, and ease in learning the system originally and assimilating advance features. This is attributed to the visibility of objects and actions of interest, the direct manipulation of objects of interest, rapid performance, incrementation and reversal of actions.

The reported enthusiasm of DM users (e.g. Chin 1984, Jones 1990) contrasts with the experience of users of command language-based interfaces. An example is the UNIX operating system, which is an example of a 'conversational style' interface, containing a number of non-natural language commands. Norman (1981) describes the UNIX interface as 'counterintuitive, inconsistent, and difficult for people to learn'. They will also have the prospect of deciphering coded system feedback, particularly after errors. This accounts for the difficulty that many users face in using such systems. The example, of moving files on a command-based system emphasises this point. The user has first to be aware of a non-natural language term(s) meaning 'move'. The user will also have to remember the coded names for the file and directories. The user will have to construct a four- place predicate, on a screen with, perhaps, nothing on it except a prompt and whatever s/he chooses to type. Only then will the operation take place.

In the file-moving example there is a considerable gulf between the thoughts and goals of the user and the description of the system which s/he is having to deal. This gulf is both in terms of structure and appearance. Users are faced with the task of translating their own representation of a task into the language of the interface. By contrast, DM systems allow the user to feel in greater command of the dialogue, and in greater control of the system.

The most well-known phrase describing the nature of DM is the principle of WYSIWIG, or 'what you see is what you get'. Hutchins et al (1986) describe 'the feeling of involvement directly with a world of objects rather than of communicating with an intermediary'. The removal of the 'intermediary' appears to be a key element. Task performance involving familiar implements such as a pen and paper, or tools such as hammers, planes and saws, allow the task performer to concentrate attention on the task-goal. A letter-writer can concentrate on the subject matter of the letter. A DIY tool user, can concentrate on the object that is being crafted. The tools are only the focus of attention if they break or malfunction. The examples display a fundamental tenet of cognitive science, namely the synthesis between human and environment. This application of this principle is applied to HCI by what Nelson (1990) describes as the 'principle of virtuality'. This describes an interface that gives the user a sense of being involved directly with a representation of the real-world. Rutkowski (1982) also describes the synthesis of user and environment with the 'principle of transparency'. This describes the user as applying intellect directly to the task as 'the tool itself seems to disappear' (Rutkowski 1981). Hutchins et al (1986) refer to this qualitative phenomenon as 'direct engagement'.

Direct engagement at the interface may be seen as the situation where the user (whatever his/her background) performs as an expert performs when presented with a familiar task. Tasks and their associated sub-tasks and routines are recognised and performed easily. The user is able to concentrate directly on the task that the interface is mediating. The user is not conscious of the interface.

Direct manipulation should produce expert behaviour in the sort of users are commonly referred to as computer naive. In other words, the novice should exhibit expert-like behaviour. The user should be able to acquire skill quickly using declarative knowledge, as described by Fitts and Posner (1967) and Johnson-Laird (1983). Work by Carroll (1984) suggests that DM systems may facilitate inductive learning in which the user can apply a minimal amount of declarative knowledge, with a small amount of training. The novice user cannot call on 'device expertise' in computer system use. Therefore the virtual world must, in some way, tap the task-performing ability formed from users experience of other tasks. This implies that its appearance and functionality, must represent real world task situations in a way that stimulates rapid expertise.

2.2.3. Achieving the Qualitative Contrast

Theoretical work by Norman (1986, 1989) further describes the nature of 'direct engagement', and the barrier that the medium may impose. Achieving direct engagement can be thought in terms of bridging two gulfs, those of execution and evaluation.

The gulf of execution can be illustrated by considering task execution in a typical command based interface. The user may, for example, wish to move an account file from one directory to another. This is a simply expressed desire and is easily visualised as a real-world task. This will be something like two drawers visible in the foreground, one with the relevant file in (instantly recognised as the relevant file). The action is a simple lifting the file out of one drawer and into the other. It's equivalent in a command-based system is somewhat less straightforward, as the earlier example described.

In the above example there is a gulf between the thoughts and goals of the user and the description of the system with which s/he is having to deal. This is true of both structure and appearance. The user is faced with the task of translating their own representation into the language of the interface.

The gulf of evaluation is also evident in this example. This is defined as the 'amount of processing structure required to establish whether a goal has been achieved'. Again this refers to the difference between the envisaged goal state as it is represented 'in the head' and the interface representation. In command systems the results of user input behaviour will often remain unclear.

2.2.4. Direct Engagement in a DM System

In a good DM system the gulfs of execution and evaluation will be closed (Norman 1986, 1989). The 'move file' task, for example, can literally be the moving of a file. The user can drag the file from one location and place it elsewhere. The user is, therefore, performing the task that s/he had conceptualised. The task seems straightforward and familiar, and is performed in the 'normal' way. The 'closure' occurs because the design and the users model have been brought together. Evaluation is similarly straightforward. The user simply observes the screen and assesses whether the file has been manoeuvred successfully. There is nothing to confound the feeling that the the task is performable and comprehensible. Five crucial user-centred components of a good DM interface can be identified:

1. The appearance of the interface stimulates recognition, and recall of appropriate task knowledge

2. The structure of the domain as it presented is consistent (though not necessarily identical) with the users own conceptualisation

3. Sequences of actions are consistent with user expectations, and recognisable as such, with the user knowing what action to perform by observing the screen

4. It is easy to recognise not only what actions are necessary but also how to perform them

5. The screen representation supports the user's conceptualisation of the goal state, and intermediary states towards it.

This, of course, is to be achieved in a virtual world of which the user may have little direct experience. It is general knowledge and knowledge of previous tasks that is enlisted. The interface resembles, or corresponds to what is in the user's memory, formed in other domains. Therefore the way in which people use prior expertise in the performance of new tasks is crucial to the user model, and will form the basis of direct manipulation design.

The initial claim was that the direct manipulation metaphor contrasted with the 'conversation metaphor' (see Hutchins 1986) by which command-based dialogues could be analysed. However, Brennan (1990) re-introduces the conversation metaphor as a descriptor of direct manipulation dialogue. The crux of the argument was that Direct Manipulation possesses the richness of human-human conversation, unlike other types of interface. This is despite the absence of textual dialogue. The first point is that meaning in conversations is often expressed though supplementary dialogues such as gestures. Another related point is that dialogues, particularly where tasks are involved, tend to involve artefacts which mediate the conversation. Phrases such as 'put that there' are used along with appropriate gestures at relevant objects.

Another aspect of the conversation metaphor is the collaborative nature of dialogue. Clark and Schaffer (1987, 1989) refer to the 'collaborative model' of conversation. In this model, dialogue is viewed as a sequence of presentation and acceptance acts between the participants. One participant presents a propositional statement. The other responds with an indication of understanding or attitude, or by an appropriate action (acceptance act). This may be a nod or a frown, or a more assertive verbal response. The acceptance act may, in turn, serve as a presentation act for the other participant. The 'collaborative model' seems to reflect the description by Draper (1986) of input/output couplings in DM dialogues. Draper describes DM as allowing the users to operate upon output representations and use them as input back to the system.

The theme of dialogue as a set of conversation-style 'turns' is further advanced by Payne (1990, 1991). Payne uses example dialogue sequence from MacDraw to illustrate the principle. The sequence of actions required to open a new file is described as a series of interactive turns. Each unit of interaction is composed of 'presentation and acceptance' dialogue acts. System responses to mouse movements and button presses are, in the first instance, acceptances of user acts. They simultaneously serve as presentation acts, cueing further response from the user. Payne (1990) uses this metaphor to describe a parsing error in which the user fails to interpret a system response to a mouse movement (on to a tool option). The user interprets a shimmering option highlight as acceptance of a selection option. The system has made an 'acceptance' of an unconfirmed selection (the shimmering presenting a cue for a mouse depression by the user). The user commences a drag action, whilst the system reverts to the tool option that had been selected before the current action. Therefore, the user starts a drag action with the wrong tool option selected. This description of dialogue further elucidates the description by Draper (1986) of 'inter-referential' DM input and output.

2.2.5. Typical Components of a DM Interface

The space of possible DM interfaces is considerable. The nature of the design may vary with the type of application, how the domain lends itself to representation by a visual metaphor, and the scope of the domain to which it is applied. However, these interfaces are typically defined as WIMP interfaces (windows, menus, icons, pointers). These elements of a DM interfaces are now discussed in turn.

The use of windows at the interface allow users the equivalent of parallel applications programs running simultaneously. As Dix et al (1992) puts it, 'users can direct their attention to the different windows as they switch from one thread of work to another'. Users are able also to manipulate the size and shape of windows. The technique allows more complex and fragmented elements of tasks to be supported. For example, cut and paste facilities are supported by a 'show clipboard' feature which allows the user to see what has been cut (and will appear as the result of a paste). This is achieved by displaying a small window which displays the content of the buffer.

The use of icons at the interface is a crucial component of non-textual dialogue. Icons may, for example represent closed windows that are available to open, operations available for selection, or changes of operational mode (such as changes in cursor mode). They are typically divided into two design styles, pictorial and symbolic, although other designs are possible. Examples of pictorial icons include the Macintosh 'wastebasket' icon representing the location of a delete buffer, and the 'mailbox' icon representing incoming messages on an e-mail package. Symbolic icons may be directional symbols (e.g. fast forward buttons, or more abstract representations. Pictorial icons are potentially able provide a direct representation between task concept and device feature. Rogers (1989) claims that the degree of icon effectiveness is dependent on the directness of the mapping between an icons physical form and its referent. It is argued that the learnability of icons is dependent on the representation, and that pictorial icons are likely to be more effective. However, Green and Barnard (1990) suggest that wider issues of array design and positional strategy are also important.

Menu operations are less direct than other DM operations in that they are selectable options which cause a change in the system state (Paap and Roske Hofstraand 1988). Menus found on typical DM packages tend to be pop-up or pull-down operated and frequently use textual representations as identifiers. However, some menus are simply given textual headers, and are represented iconically (e.g. Fill, Lines and Pen menus in MacDraw). Other types of design may also be regarded as menus (although their design contrasts considerably with the traditional notion). Palettes are used to make

the alteration of cursor modes easier and more visible. The principle of scanning from a range and selecting an option is much the same.

Other techniques that may be employed include dialogue boxes and radio buttons. Dialogue boxes are designed to show the user where to enter information. They are presented to the user in a manner similar to form-filling interfaces such as those described by Ogden and Boyle (1982) and Pakin and Wray (1982). Radio buttons present the user with a limited range of options which can be selected using the pointing device. Both examples are generally presented in sub-windows called in response to command actions on menu or icon features.

2.2.6. Some Examples of Direct Manipulation Applications

Word processors that have a significant DM content are now commonplace. Examples include Microsoft Word, EMACS and MacWrite. The advantages are both in execution and evaluation. The user is able to manipulate the cursor using a mouse, and call editing functions using the mouse. There are facilities for viewing pages of documents, and the text is generally presented in the form in which it will print. The results of action are immediately displayed on the page.

Draw packages such as Paint and MacDraw amply demonstrate two of the fundamental aspects of DM interfaces. Objects are created and edited directly by the user. This may include dragging a shape to alter its dimensions, or moving the positioning of various shapes on-screen. These are performed by directly pointing to, and dragging the target objects. This is a very direct operation on the objects of interest by the user. Also, the user is continually tracking responses to input by watching the objects of interest alter their locations and physical dimensions. This type of continuous cognitive feedback was found to be highly effective and time-efficient in studies by Te'eni (1990).

The desktop metaphor is one of the best-known and applied DM examples (Smith et al 1982, Bewley et al 1983). Information retrieval systems using DM graphics have also been reported. One example of the latter is the 'Information Visualiser' (Card et al 1991). The LISA desktop package was found to be considerably more popular with sample users than a form-filling interface (Carroll and Mazur 1986).

A number of contemporary video games employ DM interaction design. These include 'space invaders', and a many more sophisticated descendants. For example, computer golf mimics the the wielding of a golf club with a spring-action input device. It also mimics the variable conditions and constraints of real golf, including cross-winds and green conditions.

Computer-aided learning has also applied DM techniques. One example is Interactive Physics which demonstrates physics concepts on the screen in an interactive format. For example, users are able to lift and drop objects to see how their behaviour is affected by force, gravity and so on. A navy training simulator reported by Hollan et al (1984) used gauges, dials and knobs which students could directly manipulate. Another similar example is the Alternative Reality Kit (Smith 1987). Again the attempt is to teach physics by simulation, with some buttons having mass velocity, and all controls behaving as in the real-world. The technique is seen as an example of learning by doing in a virtual domain.

2.3. Models of the User

2.3.1. Introduction

The history of user modelling research reflects the developmental history of user interfaces. The original user-modelling attempted to comprehend the nature of command-based interaction. Conversely, more recent work has reflected fundamental differences in the nature of interaction, dependent on the type of dialogue that the system offers. However, some aspects of earlier user modelling research remain pertinent. The following sections describe, in loose chronological order, the development of user modelling research and its move towards describing display-based interaction.

2.3.2. Evaluative Models

In general, evaluation models do not posit well-developed theories about the nature of task knowledge, and what this implies as design criteria. Nevertheless, this class of models develops the notion of successful task performance dependent on a synthesis of interface representations of task (specifically task structure in these cases).

The Keystroke Level Model (Card et al 1981) attempted to predict performance times for sequences of user actions. The predictions are derived by giving parameters for the capacity and processing speed of users' cognitive, perceptual and motor processors. This analysis has some utility in analysing usability, and provides an account of separate but interconnected mental processing. However, the model assumes expert behaviour, and does not account for the possibility of user error

(whether user or system caused). As such it is unable to account for how interface or task-design may cause errors.

The GOMS model (Card et al 1983) describes user behaviour as composed of four key elements, namely goals, operators, methods and selection rules. The users knowledge, in this interpretation, is the encoding of methods to achieve goals. Goals decompose into methods, which are themselves composed of operators, and selection rules for their use. This gives an 'idealised' account of user behaviour. It allows the user of the technique to input a task and decompose it into a fine-grained sequence of individual operations, accounting for sub-tasks, as well as analysing the 'keystroke' level. The model, however, does not address display-based action, and its utility for display-based interfaces is therefore limited. The analysis embodies the notion of user's task models as deterministically planned. In other words, it does not allow for the influence of situated action (see Suchman 1987). Neither does it account for errors and their consequences.

The inadequacy of the GOMS model for DM is emphasised by reference both to subsequent practical and theoretical developments. The practical study conducted by Mayes et al (1988) provides evidence that users do not use deterministic planning on display-based interfaces. They studied the recall performance of experienced users of MacWrite. MacWrite is a display-based word processor which uses a number of selectable menu-commands. It was found that the subjects could not remember the names of a number of features that they routinely selected and used. Therefore, it is unlikely that their performance could be based upon whole sets of internalised information. The findings strongly suggested that the display was providing information which determined task performance. Recognition seems therefore to be a more significant catalyst for user action than recall. These findings are further backed by the findings of Payne (1990). In a further study (O'Malley and Draper 1992) it was found that, while details of the display's appearance were not retained, users did internalise spatial information.

The experimental findings of Mayes et al (1988) is backed by other theoretical work. Work by Payne (1990), and Norman (1988) among others reflect the need to see user's models of systems as distributed between subject and artefact. Whilst it is clear that some information is held in long-term memory, other information is 'left in the environment'. The major thrust of the theoretical developments is that knowledge (in the context of DM interaction) is distributed rather than wholly stored in memory. Therefore the GOMS model lacks a fundamental element. It lacks an account of how the display influences the user.

The GOMS model does provide indicators as to how investigations of the user on DM may proceed. The GOMS procedure works using selection rules, which influence the user's application of methods on the device. Whilst it is clearly inappropriate to think of these as determined without reference to the display (as the GOMS model suggests), it may give some clue for future developments. The user must have some criteria for selecting operations, and this may be the role of the display.

Cognitive Complexity Theory (CCT) (Keiras and Polson 1985) developed the notion of GOMS as structured knowledge held in long-term memory, which is accessed in task performance. The theory held that task complexity could be identified by breaking it down in terms of the number of production rules (see Anderson 1983) that are required in its performance. Users' operational knowledge is represented as a set of productions. Device knowledge is captured in Generalised Transition Networks. Complexity is measured by considering first the number of productions in the user's notion of how to perform an individual task. Secondly, the number of rules required in order to perform the task on the device is considered and compared to the user's notion of how to perform an individual task.

As with GOMS, CCT provides no notion of how display factors influenced task performance. The method assumes, in the case of each action sequence, that the user can find needed features and perform needed actions. It is simply the number of those actions that are considered. Knowles (1988) argues that the approach is diminished by its failure to analyse qualitative aspects of complexity, such as the way that the domain is represented.

CCT contributes an important notion to user modelling, that of structural matching between user's task model and device task. The user brings to interaction a mental model which will influence their attempts at performing tasks on the device. The system design embodies a model of the user (the system model). Broadly, complexity is measured by comparing the number of steps in the user's (device independent) task-model to the number required to perform the task using the system. A design is at its least complex when the number of steps taken to perform it matches the number of typical task-steps.

2.3.3. Grammar Models

Task-Action Grammars (TAG) (Payne 1984, Payne and Green 1986) model user memory for the interface language of computer systems. The method adds notions of semantics in use of commands to earlier attempts at capturing syntactic structure of command languages (e.g. Reisner 1981, 1982). TAG builds on the claim made by

Johnson et al (1984) that there are task knowledge structures in long-term memory. This is an important development as it supports consideration of how new tasks are learned and performed using previous knowledge. This includes knowledge of tasks with similar elements or features. There is a reference to task independent knowledge of such things as pointing and typing, and fundamental knowledge about spatial relations. TAG seeks to explain why some command languages are more consistent and therefore easier to learn. The claims behind this approach provide useful insights for the study of DM systems. One claim is that user uses the semantics of the task world to capture structural resemblances between the task on the device and the user's model of the task. TAG introduces the 'known-item' function which accounts for the pre-existing meanings of action elements. This describes the way in which representations of meaningful labels for the user cut down the amount of learning of the required. However, despite describing the knowledge required to map tasks onto device actions, it fails to account for the role of the display.

Howes and Payne (1990) develop the TAG notation in order to describe the role of displays. They describe a notation called D-TAG, which contains modifications to the original notation. Whilst they focus primarily on menu-based systems, the description is pertinent to DM interfaces. The 'task-item function' is replaced by a function describing the scanning and matching of display items to task features. This is called the 'display-item function'. It describes the way that the display is scanned for task-relevant features. The user scans a set of candidate interface objects. The display representation (e.g. menu name, icon) suggests a semantic definition of the object. Each definition is compared to the users current task. The best match between a display object and a relevant task feature prompts a selection by the user. For example, if a pull-down menu is being scanned, the options on view are subjected to the matching process. The chosen item is selected on the basis of its apparent semantic attachment to the user's task.

The notion of scanning and matching of options to task-features is useful for the modelling of DM interaction. This emphasises the notion of the display as an indicator of how to approach novel tasks on the device. The principle may be extended to the direct manipulation of objects. Whilst the scanning of options is less likely to be relevant in such a context, the matching of task features to semantic information on the device may serve as broad description of how users decide to act. The semantic information on-screen may still be a set of options (i.e. the space of possible actions). Also, the 'display semantics' referred to will often be the feedback from previous action. This accommodates the notion of 'inter-referential input/output' described by Draper (1986).

2.3.4. Prescriptive Techniques

Another class of modelling techniques were designed to inform the design process, rather than to evaluate systems post-hoc. These models translate user modelling knowledge into techniques for representing the domain in a user-orientated way.

Task Knowledge Structures (TKS) is the most notable attempt to establish the true nature of task knowledge and performance, and to make design recommendations on the basis of this knowledge (Johnson et al 1988). The actual design framework that emerges is conditioned by the task from the users point of view. The theory is used in the KAT knowledge capture method (Johnson and Johnson 1991, Johnson 1992). This method attempts to elicit the structure of tasks, the relative importance of task elements, how task knowledge is stored, when it is accessed, and how it is utilised.

TKS theory defines the approach as 'a summary representation of the different types of knowledge that are recruited and used in task behaviour'. The belief is that task knowledge is represented in long-term memory in conceptual or generic structures. The structures have levels of decomposition, similar to that which occurs in GOMS related models. The process is to identify the knowledge that is in a TKS, and to establish the generic elements (and therefore the abstract structure of the task) as they are represented to users. When this is established, and a completed model composed, design recommendations (or a design model) are facilitated.

Generification is a crucial component of the process. An important feature of task knowledge is the differentiation between structural entities and other incidental details. This is similar to the claim of Schank (1982) that knowledge of repeatedly occurring events are stored in long-term memory units (memory organisation packets). Memory distinguishes between relatively incidental information related to tasks and typically representative information. The notion of representative elements associated with concepts is similar to that proposed by Rosch (1975), and Rosch et al (1976) for natural categories. Further support comes from Galambos (1986) who found that recognition of event types triggers the use of task-knowledge structures. In particular, Galambos found that knowledge of the order and sequence of important events are used in understanding and guiding predictions about new events. Structural entities are seen as the essential components of the task as it is represented in long-term memory. The structure of a task, with its spatially and temporally ordered elemental relations, is held in long-term memory and applied in new situations. Gentner (1983) defines analogical reasoning as comparing entities whose structural and relational properties (rather than superficial features or incidental properties) are identical. Gentner cites the analogy 'an atomic structure is like the solar system' by way of example. This refers

to the fact that, in both cases the relationship between a central body, and smaller orbital bodies are abstractly the same. The relational structure in one domain is applied in the other, and it is just this structure that the generification process captures. In describing a domain task via a virtual world, a knowledge of the analogical structure, and a suitable description in a commonly known domain can provide valuable explanatory power. The distinction indicates to the designer the most economical way of describing tasks, in terms of the number of necessary steps that have to be included.

TKS provides a useful analysis of task content. There is consideration of constraints within tasks, which provide clues in how to assist navigation through task performance. Users accessing of task structures implies that they will have expectations about what will occur and when. Some behaviours are carried out together, or in some cases one will generally follow another, and so on. TKS gleans information about what these expectations are by finding the structure of the task, and and the spatial and temporal relations that must hold. The design of constraints and guides can proceed using this information.

User models of task-structures, and plan-based task performance is dealt with by TKS. However, DM designers have to consider the use of metaphor and general visual cues for task performance that is not plan-based. Users in this type of case will be searching for 'the next action' rather than any better-formed expectation. In such cases the crucial information is not about abstract structure, but about how screen images will cause certain types of behaviour. This need is not served by TKS.

TKS provides useful information about the structure and content of task knowledge, and provides a useful elicitation methodology. However, this is not enough, on its own, to facilitate good DM design. Logical knowledge must be translated into specific physical interface features. There must be information on how objects and actions are represented, and what stimuli cause their satisfactory use. Although TKS provides a baseline analysis, it does not provide vital information which is necessary for the production of the appropriate system image.

2.3.5. The Display-based Approach

The development of D-TAG (Howes and Payne 1990) reflects the role of environmental prompting in user models. This development of TAG was the integration of research into the nature of environmentally-based action with user-modelling developments. Suchman (1987) describes environmentally-determined action specifications as 'situated actions'. This stresses the distinction between recall and recognition in task performance. The user models described above emphasised

recall-based action, whereas DM and other display types appear to work by prompting recognition. Furthermore, the experimental work of Mayes et al (1988), O'Malley and Draper (1992) and Payne (1991) suggests that much of the information for task performance is not stored in long-term memory. The user stores sufficient knowledge for task performance and no more. Nickerson and Adams (1979) performed a study on U.S. citizens knowledge of the features of common coins, finding that inessential features were not remembered. O'Malley and Draper (1992) found the same trend when testing subjects' recall of details of keys that they used and carried.

These findings described above help to characterise memory storage as minimalistic, retaining just enough for performance. This emphasises the need for an ecologically-based approach to HCI, certainly where DM style interfaces are employed. The phrase 'Ecologically-based HCI' is used by Payne (1990, 1991) to characterise HCI as problem/resource driven. The thrust of this approach is that the essence of the user models should be the interaction between user and environment. The user cannot be understood without understanding the contribution that the environment makes to problem solving and task-performance. This contrasts with the previous philosophy, where task-models of the user are proposed first, and then applied to interaction (see Carroll 1991).

Norman (1987,1989) describes a seven-stage model of action. This splits action into goal formation (one stage), execution (three stages) and evaluation (three stages). The claim is, broadly, that task-knowledge is of a general nature, similar to the claim of Johnson et al (1988). The precise nature of a task is specified only when the environment has been scanned. The attributes of the current environment show how the generic task may be actually performed.

The approach of Norman (1986,1988) implies that there is a process of mapping between the users' prior knowledge and the physical dimensions of whatever problem is presented. In task-performance this is a mapping between (generic) knowledge related to a task, and the dimensions of the current task as perceived in the environment. This theme reflects previous work on internal/external task mappings by Moran (1983). The theme is further advanced by the 'Yoked State-Space Hypothesis' (Payne 1987). This states that 'the user of any device must construct and maintain two separate state spaces, the goal space and the device space, and some semantic mapping between them'. The device space is the environmental constraint or affordances that determine what the user needs to do. The device space must be capable of representing all the states in the goal space.

The yoked state-space model refers to the mapping between task and device space. It

can be surmised that the display must provide representations that map to representations of the device space. This theme is taken up by Tauber (1990) in the ETAG model (Extended Task Action Grammar). ETAG is a description of the user's task language. It refers to the 'user's virtual machine', a description of the knowledge needed to understand the task-related work of the machine. The decomposition of tasks (as represented in the user's task-space) are described in terms of the device's descriptions of steps. Performing tasks on the device is seen as a process of mapping the task to 'conceptual objects and conceptual operations referred to on the system's side of the task' (Tauber 1986).

Given that a DM system is rich in icons, names and object representations which effect the selection of objects, ETAG refers to relevant criteria. Further elucidation is required, however, of how the mapping of task and device space develops through the experience of using the system. Shneiderman (1986) suggests that learning the device-space is a process of exploration. Novice users tend to generate explanations from implicit notions and implicit beliefs (Turkle 1984) and may make ad hoc explanations for events during interaction (Mack et al 1984). This theme is developed by Lewis (1988) who analyses the nature of learning by doing, a key theme in the use of DM interfaces. In particular, Lewis focuses on the generalisation of procedures. This embraces the theme of characterising the nature of a set from examples of that set. This echoes work by Roth and Frisby (1976), Johnson et al (1988), Rosch (1975,1976) and Winston (1982). Winston (1982) describes the generalisation of procedures using evidence from previous examples. Lewis cites this as a key to learning of system operations.

Lewis (1988) describes contemporary theories of procedural generalisation. In particular, he discusses theories involving generalisation from a small number of examples, or only one example. Explanation-based generalisation (Mitchell et al 1986) describes generalisation as a proof, within a specified domain theory, that an example belongs to a set. This allows generalisation from a single example. Explanation-based learning (DeJong and Mooney 1986) describes the analysis of an example procedure as embodied in a set of schemata that it instantiates. This allows for the explanation of causal links between the operation and its consequences.

Structure-mapping theory (Gentner 1983) describes the use of analogical reasoning in generalisation. It suggests that generalisation is made by recognition of both a structure, and a class of cases for which it applies. For example, a two-place delete procedure works for deletion of a file (type DELETE, type BROCCOLI). This example prompts the confirmation of this procedure and its application to further 'delete-file' goals. A similar use of analogy is found in the PUPS system (Anderson

and Thompson 1986). However, PUPS introduces analysis of the roles that procedural components play. This is introduced to explain how a component action from a previous example may be re-applied. For example, 'type BROCCOLI' has the role of specifying the object appropriate for an operation. 'type DELETE' has the role of removing objects. PUPS can therefore satisfy the goal of printing the file BROCCOLI by substituting 'print ' for 'delete' but retaining the order of steps.

The PUPS example assumes that nomination of the object is preceded in all cases by specification of the operation type. PUPS gives a plausible description of how users may generalise about procedures from examples. Furthermore, the generalisation embodies a notion of interface consistency, echoing the theme described in Reisner (1981) and Rubenstein and Hersch (1984). The suggestion is that users of a system make generalisations which embody assumptions about identity of operators, roles of operator types, and the consistency of their behaviour. This echoes the description of memory and event generification found in Schank (1982) and applied by Johnson et al (1988). Users recognise event (operation) types, separating the insignificant or incidental (the actual commands or actions in a single example) from the generic type.

A further generalisation technique, described by Lewis (1986) contrasts with the rigid sequencing of event types in structure-mapping and PUPS. In this model new procedures are created from separately understood components, rather than by modifying an example (PUPS) or retrieving common structures (structure-mapping). The components are individual steps, the behaviour of which is known to the user. Lewis refers to this as synthetic generalisation. Synthetic generalisation generalises only from elements of examples that are understood, whereas PUPS carries forward features that are not understood. This is referred to as superstitious, as opposed to rationale, generalisation.

Lewis proceeds to describe empirical studies of users trying to learn fictitious systems. The study shows that users may use both superstitious and rationalistic methods. The study validates two heuristics which demonstrate how users may employ superstitious methods. One is the 'identity heuristic' where users connect pairs of actions and responses that share elements. An example is the connection between the user nominating a file called BROCCOLI for deletion, and the system's removal of the file BROCCOLI. The other is the 'loose-ends' heuristic. The claim here is that users will connect an unexplained action to an unexplained result. If a user watches a demonstration and can explain all but one user action, and all but one system action, the two unexplained actions are assumed to have a causal connection. Lewis provides empirical validation of these heuristics. In doing so he proves that rationalistic models of human learning (explanation-based models) do not completely account for example-

based learning.

This has implications for research into display-based interfaces. The use of visual metaphor, screen organisation strategies and feature presentation are likely to influence the identity and generalisation of procedures. Therefore, support for user learning will be more complex than in the simple command set described in Lewis's delete and print examples. It seems that the 'identity' heuristic may have a strong influence on novice users' attempts to explore a DM interface. In DM the user may have to identify menu options, icons or a variety of object states. Many of the execution and evaluation actions in DM do not involve textual dialogue. Users are likely, in some sense, to try to elicit designers' models during learning. Bullock et al (1982) refer to this as the 'mechanism principle'. The principle claims that causal attribution is more plausible if there is a mechanism that could mediate the causal connection between two events. Empirical study by Pazzani (1987) suggests that this strongly influences human reasoning. A computer, particularly a DM package, offers the user a chance to infer causal and procedural connections through implicit representation of dialogue.

The 'mechanism principle' will also influence the use of the 'loose ends' heuristic. In an artefact (as opposed to a natural system) it may be assumed that some purpose is attached to each action. Therefore, the user may link current system action to a prior user action in explaining an unexpected system response. However, the limited capacity of user memory for prior action casts doubt on the use of this heuristic. It may be that a number of actions are simply forgotten, and not available when the user subsequently looks for explanations.

2.3.6. Metaphor and Analogy

Mental model formation (Johnson-Laird 1983, 1988), and in turn the learning and use of systems, has increasingly been linked to the interpretation of display items at the interface (Suchman 1987, Norman 1988). However, it is difficult to establish what mental model of a system a user has (Wilson and Rutherford 1989). Some of the more recent developments in user modelling have acknowledged the importance of display factors in user performance and mental model formation (e.g. Karat and Bennett 1989, Kellogg 1990). This links to the theme of binding the attributes and semantics of the task-space to the device space. Similarly, there have been developments in understanding how menu names work and how icons may effect recognition (Gittens 1986, Howes and Payne 1990). This section explores the wider context of how the design influences users' mental models, and the binding of task-space and device-space.

The use of metaphor has been been cited as a principle feature of DM interfaces (e.g. Erickson 1990, Carroll et al 1991). A general system metaphor may be visible to the user. However, a concrete metaphor will not necessarily be available to the designer (Sutcliffe 1988). The term metaphor has a wide scope, allowing a number of imprecise connections. A metaphor may work by having some superficial property which binds a task concept to a device concept. It may also work in a similar manner to structure-mapping theory (Gentner 1983), where structures are recognised.

Metaphors are generated and used spontaneously during learning (Carroll and Mack 1983, 1984). However, there are contrasting approaches to understanding how they are used. Carroll et al (1988) describe three approaches to understanding how metaphor works. These are operational, structural and pragmatic approaches. Operational analysis studies the behavioural effects of a metaphor. This may be used to increase user understanding of a system, and also to enhance user confidence. For example, Foss et al (1982) describe the use of office storage and retrieval as a metaphor in a text editor. The metaphor was used both at the interface and in a brief training package, in which the system was described as a set of tools for performing 'familiar' office tasks. The training package was shown to improve user performance in empirical tests. This was partly attributed to the effect that this metaphor had on user confidence and perception of the system as a tool.

Structural approaches to metaphor (e.g. Gentner 1983, Douglas and Moran 1983) stress the structural correspondences between base and target domains. Primitives and relations are described for both base and target domains, along with mappings between them. This approach reflects the themes of mapping between the task and device domain, described by Moran (1983), Payne (1991) and others. This approach allows some useful related concepts to be described. These include 'base specificity' which refers to the extent that the metaphor structure is explicitly understood. This is a useful concept in assessing the adequacy of a system metaphor's coverage of a domain. A further concept is that of 'clarity' which refers to the directness of individual base/target correspondences (e.g. one to one, one to many).

Pragmatic approaches to metaphor recognise fallibility and incompleteness in mappings between base and target. Recognising such 'imperfections' as incomplete or imprecise correspondence between base and target, the pragmatist may exploit advantages that are afforded. For example, Smith (1987) used a customisable 'law of gravity' in the ARK tutoring system. This maverick feature was presented alongside more literal domain/device mappings. The use of composite metaphors has been found to help users generate more varied explanations of system behaviour (Williams et al 1983).

Mountford (1990) describes the use of various metaphors in Macintosh applications. She cites the correspondence of appearance and behaviour of interface objects to real-world counterparts as key factors in system usability. However, there is less understanding of the range of utility that metaphor concepts provide. Lakoff and Johnson (1980) claim that metaphors do not imply a complete mapping of every concrete detail of one object or system onto another. Halsasz and Moran (1982) argue against the use of base/target analogy for detailed reasoning about systems, limiting the role of metaphor to conveying individual points. A further complication is that metaphor may influence interaction at different levels. A metaphor embodied in a single icon may influence only interaction with that single feature. However, the general system metaphor (e.g. the desktop) is likely to have a wider influence.

Whilst virtual representations of domain objects are important interactive catalysts, there is dispute and debate about their actual and ideal function. Nelson (1990) criticises the fact that metaphors tend to overgeneralise. For example, the wastebasket is a well-known feature of the desktop metaphor. The functionality of the feature is either deletion (which requires an extra step) or storage in a remote buffer. The metaphor, however, strongly suggests that the feature is purely for 'waste disposal'. This is held as an example of how a function may be given an inappropriate or limiting representation in order to fit a general metaphor. He points out that a high-level metaphor can be comprehended at a number of different levels. There may be a level at which it fits the device space well. However, further extension of the metaphor may well diverge from the target system functionality. The problem of metaphor coverage raises further questions about the nature of its function. The wastebasket example appears to suggest that metaphor indicates functional identity. It is possible to think of other metaphors (e.g. lasso) that make a clear statement about how to operate the feature. Also, as Lewis (1988) shows, users tend to respond to perceived group resemblance between objects and functions. User perception of domain objects as 'group members' may be triggered by interface metaphors, influencing user beliefs about their identity and operation.

2.3.7. Conclusions from Study of User Models

The theme addressed by contemporary user models is matching system functions and user knowledge held in long-term memory. In the first instance, models have placed emphasis on complete structures in memory which are recruited and used in task-performance. However, as Howes and Payne (1990) observe, these models are unable to characterise display-based interaction. Display-based interaction demands a more 'ecological' approach to user models. Task knowledge held in long-term memory is better seen as information used to interpret prompts and signals from the

display. In other words, task-knowledge must be understood as smaller chunks of knowledge, which are recruited when prompted by display features.

An understanding of the synthesis of display factors and task-knowledge is crucial to future models. They must account for how display information and techniques are used to guide user behaviour. Understanding of analogical reasoning behaviour by users and how concepts are represented in memory are also key issues. Lewis (1988) suggests that user abstraction of system knowledge is a key element of learning. However, more diverse sets of operations than those he describes typify DM interaction. A more precise account is required of how metaphor affects user behaviour at various levels.

2.4. Evaluation Approaches

2.4.1. Introduction

This section studies and compares contemporary evaluation techniques. No current techniques are specifically directed at DM techniques. However, some are motivated by theoretical and empirical work on display-based interfaces. One important distinction is between formative and summative evaluation. Formative evaluation refers to methods which influence the design process prior to the finished product's completion. Summative evaluation refers to evaluation of the finished product, with a view to validation or iteration. It can be argued that the distinction is blurred, given the availability of prototyping tools and techniques such as storyboarding (Young and Barnard 1987, Clark 1991) although some methods are clearly intended to influence design at a much earlier stage than others. The following study discusses a range of evaluation approaches. The crucial contrasts are between the nature of the techniques' application, and the theoretical thrust behind these approaches.

2.4.2. Evaluation Models

A number of the user models discussed earlier were intended for use in formative evaluation (e.g. Reisner 1981, Card et al 1983, Payne and Green 1986). Their validity and relevance as models will not be discussed here. The tools are intended to yield objective measures of complexity, based on models for the user. This section concentrates on their utility as evaluation tools. The complexity of model-based evaluation demands considerable expertise. For example, Keiras and Polson (1985) use production system notations of the type described by Newell and Simon (1972) to

describe users' task knowledge. Their formal device representations use transition networks similar to those used in psycholinguistics by Woods (1970) and Jacob (1982). In both cases, any user of the method would need a strong background knowledge of the relevant academic literature.

The utility of the class of models described above is diminished in three ways. The first is that expertise is required for its use. Hence, the problem of models acceptance by the design community remains. This problem is documented in the literature (Bennett et al 1987, Knowles 1988). Some attempts have been made to enhance the usability of such methods (e.g. Keiras 1988). Also, formal analysis of complex interfaces requires considerable time and effort. Bellotti (1990) cites this as a major obstacle to the acceptance of such methods. The third problem is related to the analysis of the models above. For a number of systems, particularly DM style systems, the methods do not address the complete problem of end-usability. The role of the display, and its potential for causing errors, is not considered by these methods. They are not equipped to diagnose display-based problems. Therefore, testing and iteration are still required after these methods have been used.

2.4.3. Taxonomic Studies

Ravden and Johnson (1989) describe the Usability Checklist approach for summative evaluation. The checklist is composed of ten sections of criteria-based questions, and a further section inviting written comments on aspects of the system. The method is designed to be usable by a considerable range of people, without fundamental modification. In particular, it may involve representative end-users in the evaluation, a need pointed to by Gould and Lewis (1985). The method poses a set of questions based on standard usability principles. The user is asked to try using the system, and fill in the checklist post-hoc. The method has the advantage of providing a list of criteria that the user may not be aware of, and potentially eliciting richer feedback than a straightforward commentary on the session, or written account of experiences.

Some potential problems exist with the checklist approach. There is an exhaustive list of questions, but some are based on criteria which may be obscure to the user, or lack a common definition. An example is the concept of consistency. Work by Reisner (1990) and Grudin (1989) suggests that the meaning of the term in the context of the user interface is far from clear.

Booth (1990), and Booth and Gray (1990) use a taxonomy to interpret user errors. Four types of error are described, referred to as the ECM scheme. These are object-concept mismatches, operation-concept mismatches, object-symbol mismatches and

operation-symbol mismatches. The term 'objects' refers to anything from a package to a single icon. The operation is an action which is performed on object(s) within the system. A concept is an object or operation whether represented mentally (by the user) or physically (by the system). A symbol is any representation of a concept (e.g. word, icon, figure, shape) employed by either user or system to represent a concept.

Critical incidents are first examined by identifying the object and operation. The next action is to link both to a concept and symbol. The final stage is to describe the position of the mismatched element within the dialogue failure. This final stage investigates the role that the element played in the dialogue failure. Booth and Gray (1991) point out that the user's view of task and system must be elicited in order to identify concepts and symbols.

The ECM taxonomy has the problem of multiple classification. It is possible to classify an error in contrasting ways dependent on the perspective that is adopted when considering the problem. An example from the Apple Macintosh is reported by Jones (1989). The disk icon (representing an inserted disk) failed to disappear after the disk was ejected. This could be classified as an operation-concept mismatch because the concept of the operation is mismatched with the user's representation. On the other hand, the icon could be placed in the wastebasket thus removing it. This would classify it as an operation-symbol mismatch, given that the operation is possible, but not in a form that the user recognises.

Whilst Booth and Gray (1990) regard multiple classification as a problem, it may be argued that it could be an advantage. The diagnosis described is motivated towards design improvement. Multiple options represent a range of possible redesign options, all of which may be worth considering. As Booth and Gray (1990) claim, the method 'enables and directs creative thinking'.

2.4.4. User Studies

Techniques for summative evaluation involve direct observation of users performing tasks at the interface. Ericsson and Simon (1980,1983) use think aloud protocols to elicit user intentions, attitudes and interpretations of the system. Users are video-recorded performing tasks at the interface continuously verbalising their thoughts and actions. They may also be interviewed post-hoc for clarification. This technique has the advantage of linking user beliefs and knowledge-states to overt behaviours. For example, error studies using this method elicit user assumptions that may have contributed to an incorrect choice.

A major drawback of protocol analysis is that a single evaluation session generates vast amounts of data. Most of the data will have to be gathered from retrospective analysis of recorded material. It has been claimed that the phase of data extraction is extremely time-consuming to analyse (Maguire and Sweeney 1989). However, Virzi (1990, 1992) found that a fairly small number of subjects could provide sufficient data to pinpoint the most important usability problems in a system.

Maguire and Sweeney (1989) propose an alternative method of monitoring users using the 'Human Interaction Monitoring System' (HIMS). The HIMS system records all the input and output actions in the interactive sequence. The system also provides video and audio facilities. Critical events in a session such as errors and requests for help can be marked. HIMS reduces the risk of human error in analysis of user input, as interactions are logged automatically. It also reduces the transcription and analysis phase of the evaluation.

A problem with the monitoring approach is selecting data that is of interest. Maguire and Sweeney (1989) surveyed evaluation professionals finding that the aims of evaluation sessions may limit the amount of data capture required. They were typically interested in error numbers and frequency, time taken to complete tasks, and the range of functions used. However, acquiring richer accounts of user behaviour may be difficult. Maguire and Sweeney (1989) point to the problem of tracking mouse use. Not all mouse use is detectable by checking system input. For example, the cursor stops at the end of the page, even if the user continues to drag the mouse. Another problem is the difficulty of interpreting higher order actions and user intentions from low-level events of the type that this approach captures.

The Co-operative Evaluation technique (Wright and Monk 1989, Wright et al 1989, Monk et al 1991) modifies the protocol analysis technique to include intervention by the evaluator during the session. The method is partly motivated by work on 'Contextual Enquiry' by Whiteside, Bennett and Holtzblatt (1988). Users are invited to perform task verbalising continuously. The evaluator may intervene to ask for explanation and clarifications of what they are doing, and their attitude to the system. The technique is less 'natural' than pure protocols, with the inherent risk that the evaluator may inadvertently lead the user subject.

2.4.5. Criteria-Based Design Walkthroughs

2.4.5.1. Heuristic Evaluation

This method, described by Nielsen and Molich (1990), uses selected dialogue design heuristics to assist groups of evaluators. Evaluation teams are made familiar with heuristics such as 'speak the users language' and 'use simple and natural dialogue'. They are then invited to analyse an interface in the light of the list of heuristics provided. Nielsen justifies the approach by describing usability problems with the Macintosh interface (Nielsen 1990). In this study he produces convincing examples of design errors which can usefully be interpreted using the heuristics described.

The practical advantages of Heuristic Evaluation are considerable. The 'workload' in terms of paperwork, form-filling and participatory manpower are relatively light Nielsen (1992). This method is also reported as being cost-efficient, as analysis can be completed within a day. However, Nielsen (1992) concedes that at least one member of the evaluation team should be a human factors specialist. This individual may not be available to a team.

A problem with using Heuristic Evaluation for DM is that it may be difficult to pinpoint features that are error prone during use. This point is illustrated using an example user error taken from Payne (1991). In this example the user selects a palette option with the pointer (causing the option to shimmer), but fails to release the mouse button after moving it to the option. This causes the previously selected option to remain current, while the user proceeds to the next action.

So, having made an error, the heuristic 'provide good error messages' becomes relevant to the palette error scenario. How would the system tell the user what the user has done wrong? Evidence of an error is there, as the tool icon shimmers but does not become still until the mouse button is lifted. Similarly, the user's 'error' results in the highlight returning to the previously selected option. So, in that sense, the heuristic has not been infringed. DM and other display based systems tend to avoid using error messages. Indeed errors in such systems are merely schisms between device actions and user objectives, and not identifiable by the system. In system terms the user is still performing 'legal' operations. Therefore, Heuristic Evaluation, at least with the heuristics presented, is prone to misrepresent some serious errors if applied to DM interfaces. This is partly due the absence of further diagnostic guidance to establish underlying error causality. However, this does not rule out the possibility of proposing a set of heuristics directed towards the specific problems of DM design.

2.4.5.2. Claims Extraction

This approach is proposed by Carroll and Kellogg (1989) as an alternative to cognitive theory as a tool for evaluation. Existing interfaces are analysed and lessons (claims) extracted for subsequent application to new versions or new products. Claims extraction as an approach emphasises a crucial element of evaluation needed for DM. This is the emphasis on qualitative analysis of artefacts in use (Kellogg 1990). In this approach the artefact's (as opposed to the designer's) implicit model is analysed in language which reflects 'a view of the psychological dynamics of its real usage situations, expressed in terms of its design and the tasks it supports'.

It is suggested by Carroll (1990) that Claims Analysis is a substitute for cognitive theory. However, the extraction of claims involves analysis of a range of cognitive science issues, such as memory capacity and user learning strategies. Therefore, it is likely that 'claims' may be best identified in the light of appropriately applied cognitive theory, a view supported by Polson et al (1992).

2.4.5.3. Cognitive Walkthroughs

Cognitive walkthroughs are a development from 'design walkthroughs' used by software engineering teams (Yourdon 1989). Most of the work in this area has focused on specifications of 'walk-up-and-use interfaces' (e.g. Lewis et al 1990, Polson et al 1992). Lewis et al 1990 base the walkthrough structure on the CE+ model of exploratory learning (Polson and Lewis 1990). The model claims that user problem solving is based on the similarity between user expectations of an action's consequences with their current goal. The technique provides a particularly precise stepthrough evaluation of each component in an interactive sequence. The walkthrough structure is annotated with a theory based list of questions for the evaluator. In this sense the assertion of Polson et al (1992) that it can set an agenda for claims extraction seems to be a plausible one. The theory of action proposed by Norman (1986) provides a sequential structure for criteria-based questions about the design.

The emphasis on cognitive theory by Polson et al (1992) means that the technique may not be portable between types of interface. The method was designed to deal with walk-up-and-use interfaces. Many DM designs are not walk-up-and-use, requiring some training in manipulation techniques and system concepts. Therefore, the potential application of this technique is conditional on its modification to handle such

systems. The walkthough technique attempts to give an estimate the user's model of a task, partly based on sound theory. The structural element of the walkthrough based on the work of Norman (1986) is an abstract description of an interactive sequence. However, some of the user's knowledge, such as representations of task concepts or symbols, is harder to estimate. Also, DM systems typically afford alternative paths to achieve goals, something that would be hard to cover in scenario-based walkthroughs. The theoretical developments in Chapter 3 address these issues.

2.4.6. Summary of Evaluation Approaches

A key problem with using models to account for diverse designs, and knowledge states is that the scope of predictive models is clearly limited (see Simon 1988). They are unable to account for diverse factors, such as states of user knowledge, and their interpretation of display features. This is linked to the second problem, that these models are difficult to use. The more recent focus on methods that are geared to participation of user subjects addresses the first problem. This is is also addressed by methods such as Heuristic Evaluation and Cognitive Walkthroughs which attempt to make design criteria usable and accessible, whilst providing an underlying theoretical platform.

The utility of Norman's (1986) theory of action is demonstrated by Polson et al (1992). The theory provides an abstract account of display-based action sequences which is portable between types of display-based interface. By contrast, Nielsen (1992) and Ravden and Johnson (1989) emphasise adherence to usability principles rather than using models. These two approaches may not be mutually exclusive. It is conceivable that a method could pinpoint the relevance of certain principles at certain times. To some extent, the cognitive walkthrough uses this approach.

A further class of methods, including that described by Wright and Monk (1991), emphasise the need to elicit representative accounts of users' models. DM interfaces are rich in functionality, interactive techniques and cues. This increases the difficulty in accurately estimating user responses to a design. The elicitation of user protocols during actual sessions addresses this need.

A problem facing each type of method is a lack of diagnostic support for display-based problems. Heuristic Evaluation classifies problems without specifically guiding the selection of design alterations. Checklists (e.g. Ravden and Johnson 1989) provide lists of key principles, but the ease and utility of linking a list of principles to design particulars during evaluation requires further investigation. Cognitive Walkthroughs have the advantage of facilitating envisionment of the interface-in-use, but give no

further assistance in suggesting design alternatives. User-based studies, whilst collecting data which can give good insights into the user's interpretation of the system, do not provide detailed diagnostic support to help generate design improvements.

2.5. Summary

This chapter has explored attempts at modelling users, and the influence that the display has had on this type of research. Display-based dialogues, particularly DM dialogues, need dedicated models of their own, given the specific nature of communication in a graphical medium. This need is recognised by, among others, Norman (1991), Howes and Payne (1990), and Payne (1991). An emergent theme in this work is the need to model human cognition in terms of its interaction with its environment.Whilst early work seems difficult to apply, the modelling of event sequences (Norman 1986, 1988) and of learning from displays (Lewis 1988) seems to afford a theoretical platform from which models can be built. This thinking is reflected in some of the more recent advances in evaluation techniques. Cognitive Walkthroughs (Polson et al 1990) are the clearest example of user models applied in participatory evaluation. However, the utility of current walkthroughs for DM is in some doubt.

The knowledge required for evaluation includes details of what a typical user's model actually consists of. Whilst theory work provides some guidelines, a more complete picture is more likely to be yielded by studying users. The verbal protocol techniques (Ericsson and Simon 1983, Wright and Monk 1991) present an opportunity to collect such data. However, these techniques stop short of providing a theoretical underpinning for the appropriate capture and use of relevant data. Potentially data from evaluation sessions could be augmented by theoretical models to provide a richer analysis of system usability.

Chapter 3 - A Model of Direct Manipulation Action

3.1. Introduction

This Chapter describes a model of action developed from the literature survey. The model describes a number of alternative action paths that a user may take when performing tasks on a DM interface. The model embodies some assumptions that are tested in the empirical study reported in Chapter 4. In particular, the model asserts that action can broadly be characterised as opportunistic and display-led, rather than based on deterministic planning.

3.2. Theoretical structure of the Model

This model is based on a theory of action proposed by Norman (1986). Norman describes action in terms of the environment's influence on behaviour. It divides the process of action into seven stages. These stages are components of a three stage model incorporating goal formation, action and evaluation. These are now described using Norman's own example, that of switching on a light.

Norman describes a situation in which he is sitting in a chair reading, and realises he needs more light. The imperative 'get more light' becomes the goal statement. At this point he has not specified how, or by what means he will get more light

The next phase is *forming an intention*. In this case the intention is to push the button on his lamp. This in turn is translated into a more precise set of operations such as deciding whether he needs to get up from the chair to reach the lamp, or whether there are obstacles to avoid. This is referred to as *specifying an action*. This is followed by actually *executing the action*.

Having executed the action he now assesses whether the goal has been achieved. This involves three stages. *Perceiving the state of the world* is the first stage. This is checking that a perceptible state-change has occurred. The next stage is interpreting the state of the world. This is deciding whether you understand what has happened and why. The final stage is *evaluating the outcome*, comparing what has happened with what you wanted to happen (the initial goal).

3.3. The Theory of Action Applied to DM

To illustrate the relevance of the theory to DM, the theory is used to describe a delete action from the Apple Macintosh. The deletion of files on the Macintosh is performed as follows. The file is selected by a single click on the icon representing the closed file. The user then drags the file (keeping the mouse button pressed) and releases the button when the icon is over another icon, representing a remote 'delete' buffer. The user must then select a menu option in order to remove the file from the buffer, completing the deletion. The next section describes the sequence in terms of Norman's theory.

A user's goal is to delete a file. The user will be aware that this involves an act of removing or erasing a target item. This can be thought of as the *intention*. In fact, the user may also have some experience of deletion on some computer packages. This, in itself, is not enough information, but it will guide the user's interpretation of what is found when the environment (the interface) is scanned. The user will therefore expect that the operation will involve specifying the item and performing the delete action, whether this be by command or button selection. The specifics of what is needed must therefore be communicated by the interface. So the user scans, looking for an indication. In our hypothetical example the user finds the 'file' represented in the form of an icon. From this the user knows that the file can be selected using a pointer. So one stage of specification is complete. The user then has to look for an operator which will perform the deletion action. On scanning the interface further, the user observes an icon depicting a wastebasket. The user is aware that, as it is a non-static icon like the file, the file can be moved 'inside' it, using the pointer. The fact that it is depicted as a wastebasket suggests to the user that this is the correct procedure (pick up the item, take it to the wastebasket, drop it in). So the specification stage is completed with a 'drag and drop' pointer movement. The user duly performs the action, observing that the wastebasket changes colour to indicate that the item is in the correct position. The user releases the pointer, and the wastebasket bulges. The user, on observing this, realises that deletion is not yet complete, and the file has been sent to a buffer. Hence, there is an addition to action specification. The user reasons that there must be an extra action before the goal is satisfied. A further scan of the interface reveals a menu option called 'empty wastebasket'. By selecting this the user will finally delete the file. The user selects the option. On selection, the wastebasket ceases to bulge. The user sees this feedback and interprets this as a successful deletion.The example demonstrates the nature of DM interaction. It can be split into the three phases of specification, execution and evaluation. We can now describe the

three phases further:

Specification - The user may know how the action is performed in general terms, but needs to consider how the task is to be performed at the interface. Therefore the user must scan the interface for an indication of what can be done. This involves the selection of features and inferring the correct feature. The user scans the interface for a relevant feature. On selecting the feature the user reasons about the required manipulations. Typically, DM interfaces only give visual prompts, rather than specific instructions, on how to perform operations.

Execution - Execution refers to the physical user actions on the system.

Evaluation - The user compares the visible system response to input with the original goal. The user is relying on the interface to give a clear and comprehensive indication of the effects of an action. This involves perceiving information evidencing the state change, and comprehending the image in terms of the current task.

The model follows Norman's approach and assumes that much DM interaction will be environmentally determined. Therefore, what the user sees, and how the user interprets cues, are important influences on the course of interaction. Extra evidence in support of this approach is provided by Mayes et al (1988) in a study of MacWrite users' memory performance. Mayes et al. (1988) found that users' uncued memory of the system image, even from highly experienced users, was remarkably poor. Users were unable to remember information that was vital to their performance of tasks. This suggests that users' action may be determined primarily by environmental cues rather than well formed memory schema and task-action plans. This implies that an important design issue is how to support action in terms of interface metaphor, feedback and visual cues. The user recognises familiar visual signals, and is able to retrieve procedures only under this stimulus.

The findings of the Mayes et al (1988) study are further elucidated by studies of O'Malley and Draper (1990). These involved asking users to describe the characteristics of keys that they carried with them on key-rings. The study consistently found that they knew just enough information for distinguishing between keys. This would include colour, size and shape. Further details were only encoded if two keys appeared similar and further committal to memory as required. This emphasises the principle of minimal encoding. Long-term memory retains as much as is required for task performance and no more. Given this, and given the findings in Mayes et al, the role of the interface as an element of distributed memory becomes of crucial interest.

The model of action uses both the theory of Norman (1986) and the implications of Mayes et al (1988) as theoretical foundations. The model builds upon Norman's theory of action to create a description of interaction in terms of cue recognition, action and evaluation of the resulting system state. It also reflects the range of responses that users make to the visual feedback when an action is performed, including when the feedback is unexpected or unsatisfactory. Exit paths (which may occur at most points in the cycle) are also shown. The model is essentially descriptive although it does have some prescriptive power in that specific types of errors are associated with particular steps during interaction.

3.4. Description of the Model

3.4.1. Overview

The model consists of alternative paths by which users may specify actions to be performed. The path that the user takes is dependent on whether the user is able to recognise familiar features and therefore perform action. If the user cannot recognise a familiar (and relevant) feature, the need for a recognition based guess is implied. The model then describes a number of paths related to the evaluation phase of action. The path that the user takes in this phase is dependent on whether or not the action produced expected results, satisfactory results, or interpretable results. The sections below detail the stages of the model, including the theoretical underpinning of each stage and the nature of the alternative paths. The model is shown in Figure 3.1.

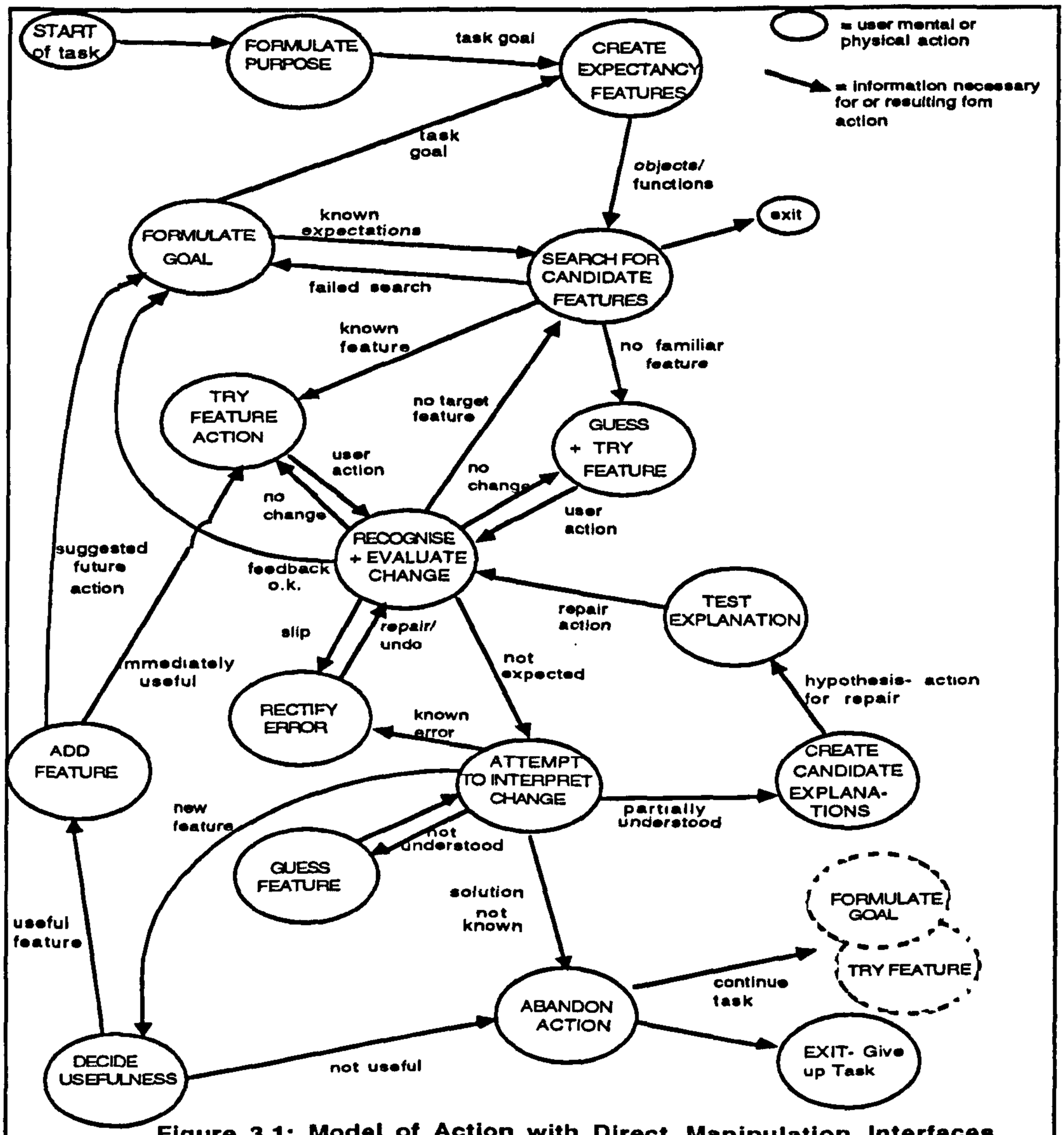


Figure 3.1: Model of Action with Direct Manipulation Interfaces

3.4.2. The Goal Formation Phase

The goal formation phase is described both for the initial task-goal and subsequent task-goals. The node 'formulate purpose' implies the selection of the first task-goal. The step 'formulate goal' refers to the subsequent formation of goals as interaction progresses. The formation of goals will be influenced by the way interaction has progressed prior to it. This is further discussed in later sections of the model description.

3.4.3. The Execution Phase

The node 'formulate goal' leads either straight to 'search for candidate features' when a familiar task is involved, or 'create expectancy of features' where relatively novel tasks are to be attempted. The 'Create expectancy of features' step predicts that users form a partial model of the objects and functions drawn from knowledge of the domain, task and device. Clearly several partial models may be formed depending on the user's experience. During interaction goals are assumed to be formed for the next intended action. Such short range goal formation is not explicitly modelled. Action is generally assumed to be triggered from the environment following Suchman's (1987) view, although more conscious mental activity is represented as a separate step 'formulate goal'.

The 'search for candidate features' node is operationalised by the users scanning the screen for an appropriate cue with which to start the task. Depending on the user's prior knowledge and design of interface cues for action, the necessary feature may, or may not, be recognised. Where a feature is known, the user can directly embark on task-action, represented by the 'try feature action' node. However, where the user is not sure the user goes to feature exploration, represented by the 'guess and try feature' node. If an explicit cue for action is not found the user may experiment with likely candidates (or in extreme cases give up).

3.4.4. Evaluation

Whether a known feature or a guessed feature is being used, the user will rely on system feedback to interpret the success of the action (Recognise and Evaluate Change). If the feedback is as anticipated, the user can formulate a new short range goal and proceed to the next action, possibly with a search if the candidate feature is not immediately apparent. If feedback mismatches with user's expectations then an error may have occurred. Error-states may be recognised and immediately corrected

("Rectify error") or further assessment of the system state may be necessary ("Attempt to interpret change").

3.4.5. Response to Unsatisfactory Action

When assessment of an unexpected effect is necessary, the user may be able to generate a candidate explanation for the error, and will take the necessary remedial action (Test explanation). When the solution is not apparent, the user may decide to experiment, possibly several times in search of a solution (Guess feature). Experimentation may suggest a candidate explanation for the initial problem, or the user may give up the task-action being attempted (Abandon action).

Abandon action may be followed by the same task-action being attempted with a different system feature, (Try Feature/action) or the user may move on to another part of the task and formulate a new goal. In extreme cases the user may give up completely leaving the task incomplete. Another possibility is that the user finds a feature that is not useful for the current goal, but does appear to be useful for some future action (Decide usefulness). This feature may be added to the user's model of the system (Add feature). If a user, for example, reshapes an object accidentally whilst performing an unrelated task, it may be remembered and utilised later on.

3.5. Display Knowledge Sources

3.5.1. Introduction

Phases of the model were linked to relevant sources of user knowledge. This was done to describe the influence of particular interface features on user behaviour. The influence of a particular design element is dependent on the phase of the action cycle, and the state of the user's system knowledge. Five types of display knowledge-sources are described. These are shown in figure 3.2.

3.5.2. System Metaphor

The influence of the system metaphor links 'create expectancy of features' to 'search for candidate features'. The system metaphor may provide an indication of the next possible action, and guide search. Typical DM packages include screen areas such as feature menus and palettes, and an operation space (e.g. the MacDraw draw-space). The organisation of the screen and its visual presentation may provide an indication of where to search. As Mountford (1990) describes, the correspondence of the visual

image to real-world concepts is key to the process.

3.5.3. Cue Semantics

This knowledge-source is specifically associated with the 'no familiar feature' path in the model. The semantic properties of an image or a textual cue provides interpretable evidence of a feature's identity. The user will scan for features that are relevant to the current goal. This is described by Young et al (1990) as a scan for items which match to the user's goal. Icons and menu options are scanned and matched to concepts from the task-space. The user will select a feature for exploratory action if the semantic connection appears sufficiently strong.

3.5.4. Cue Recognition

This refers to knowledge employed when a familiar feature is used. The feature acts as a cue to device memory. The user recognises the feature's relevance to the current task having previously used it. The study reported by O'Malley and Draper (1992) suggests that the user may need to scan and find the feature despite its previous use. Recognition of the cue may act as a prompt, reminding the user of the feature's operation. Other information, such as other features with which it associated, may also be prompted.

3.5.5. System Feedback

The result of the action is matched to the user's goal. Feedback serves as an indicator of user progress in the current task. The user looks for evidence of goal satisfaction. Feedback may reference the task space by showing a metaphor-based image (e.g. the 'message' icon moves to the 'mailbox'). However, feedback may also simply confirm the action using a system convention (e.g. a selected option becoming shaded).

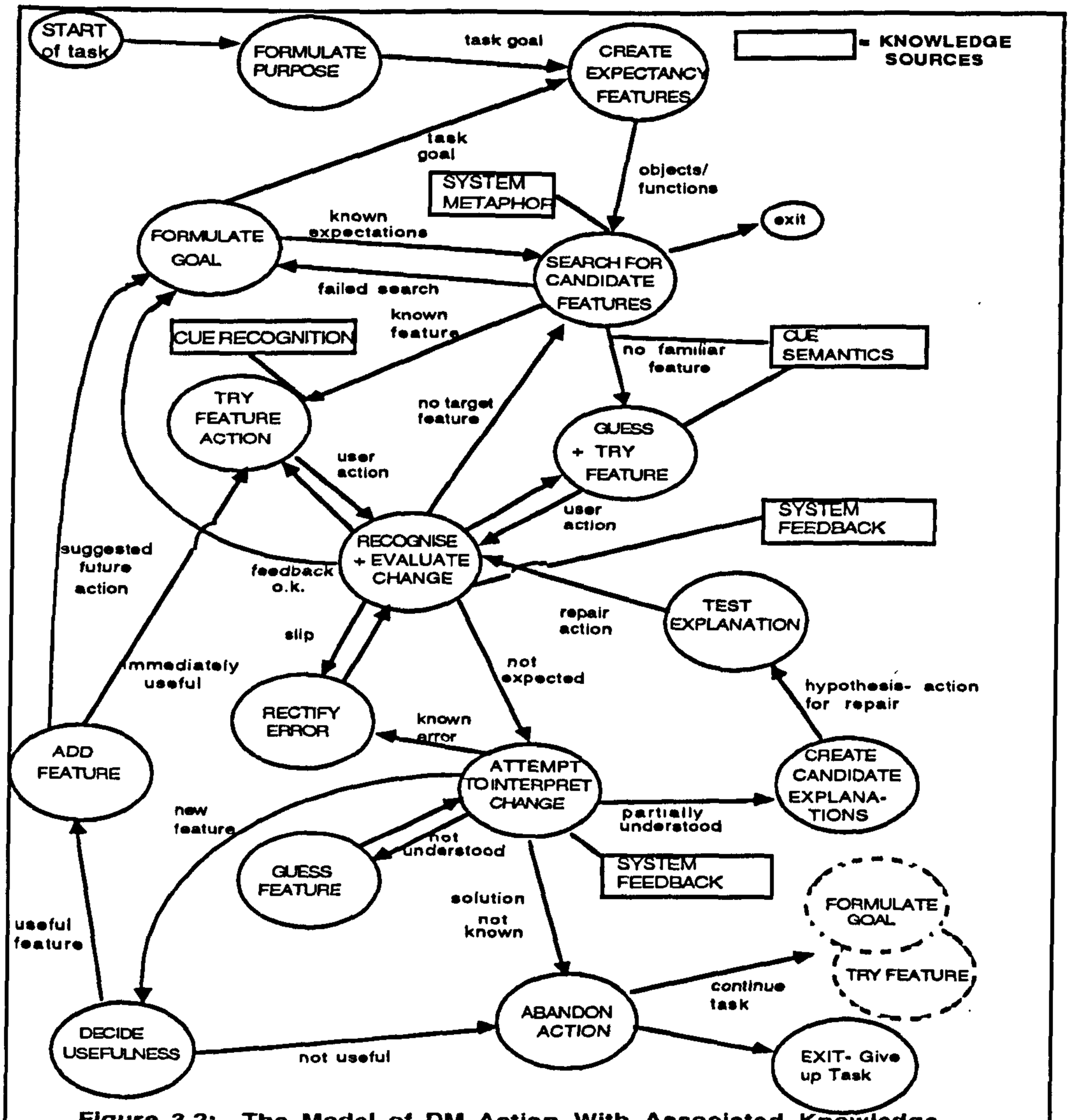


Figure 3.2: The Model of DM Action With Associated Knowledge Sources

3.6. Errors Linked to the Model

3.6.1. Introduction

Five error types were linked to the model. These errors describe ways in which the design may fail to support user behaviour. This serves as a taxonomy of system led dialogue breakdowns. The types reflect stages in the model of action. Figure 3.3. shows the error types linked to stages in the action cycle. The error types are described along with their motivation from the model.

3.6.2. Misleading Cues

Norman (1986) emphasises the phase of scanning the environment for an indication of how a goal is translated into a detailed intention to act. The environment is scanned for cues which indicate what can be done, and what needs to be done. The DM user is, therefore reliant on receiving appropriate information from the environment. In the model this is linked to the phase 'search for candidate features'. The user is searching for display items which indicate the appropriate mapping between task-space and device-space. The poor design of cues may, therefore, trigger an incorrect action and the use of an incorrect feature.

3.6.3. Ambiguous Feedback

This refers to post-action cues from the environment. The user requires confirmation that an action has had the desired effect. Again, the user is entirely reliant on the display to provide this information. On a DM system this will rarely be explicit messages. The system will simply display a state-change, leaving the user to interpret that change. The type of error referred to here is the misrepresentation of state-changes. The crucial role of feedback is described by Draper (1986), and Hutchins et al (1986). Both cite the directness of feedback as a key to rapid learning and direct engagement. Equally, poorly designed feedback may either disrupt direct engagement. Also, it may misinform the specification of subsequent action. The user may be prompted to act incorrectly as a result of misleading feedback, as feedback cues subsequent action.

3.6.4. Hidden Functionality

This category refers to elements of the design that are invisible to the user. Given that interaction is display-based rather than relying on explicit messages, the system must demonstrate the nature of the system state and system utilities. For example, the problem of modes referred to by Monk (1986) is that incomprehensible changes are made to the system state. The mode changes may often be traceable to a user action. However, the user will not be informed about the link between the change and the action that caused it. This is an evaluation problem referred to by Lewis (1988). Other examples may be features that the user is unable to find, or is simply unaware of. The limitations of metaphor described by Lakoff and Johnson (1990) suggest that signposting some system functionality may be difficult.

3.6.5. Inappropriate Functionality

Inappropriate functionality describes situations where the functionality does not behave as the user expects or needs, even though the broad interpretation of the cues are correct. This category emphasises the principle of interface features as tools. Norman's (1986) Theory of Action describes the recognition of tools or affordances in the environment. This must also imply assumptions about what these items do and the scope of their utility. This applies both to tools and the more general 'utilities' of the interface. For example word-processors and drawing packages have virtual 'paper' for writing and drawing upon. These are designed to demonstrate their utility through metaphor suggestion. The user anticipates that certain features will support a level of task performance. The functionality may be described as inappropriate if, for example, a tab-setting facility does not allow the user to achieve required settings. The feature exists but is not powerful enough for specific user needs.

3.6.6. Expectancy of an Impossible Action

This refers to errors where an action is in the user's model but is not supported by the system. The role of the system both in generating and supporting user action is examined by this category. It refers to situations in which the user is given the false impression that an action is possible, through their general expectations of the system, or experience of use. Lewis (1988) describes the system learning in terms of expectations generated by generalisations. User's memory of previous examples (the same or similar features) is a key component of system learning. Also, users' comprehension of the system metaphor will generate expectations. The metaphor provides a notion of system scope by effecting a mapping to the user's model of the domain. The user may reason incorrectly about potential functionality if the metaphor

is misleading.

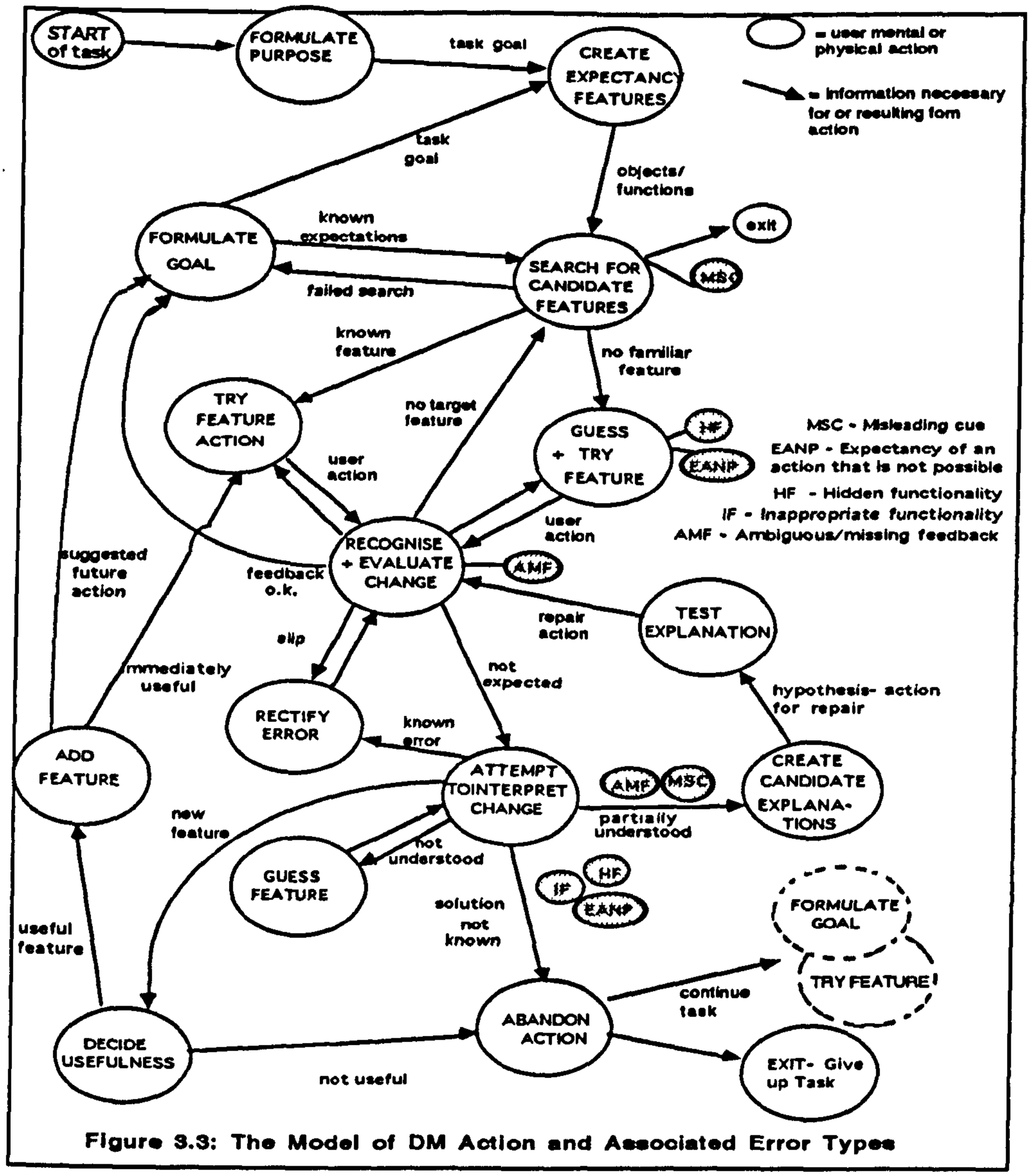


Figure 3.3: The Model of DM Action and Associated Error Types

3.6.7. Summary

The model describes action with both known and novel features, and the interface's support for search, execution and evaluation. The model assumes that the display is a key component of decision making at every stage. Users may be capable of retrieving known features of the interface from memory, but require the system image to prompt this. Where the needed features are not known the user will scan for cues to decide how and with what action can proceed. The user is also reliant on the system image to evaluate the action and specify the next action. The model accounts for the learning of new features with the parallel 'add feature' node.

3.7. Investigations of the Model

The model can be empirically investigated on three counts. One is to validate the overall premise of the model that action is display-led rather than the result of deterministic planning. This would demonstrate that interaction with DM is fundamentally distinct from command line systems, as claimed by Mayes et al (1988) and Payne (1990) among others. In command-line interfaces, sequences of task-action commands are stored and retrieved by users. If the display-led theory is valid, the DM user will perform tasks competently showing skilled behaviour without detailed memory schema for device procedures.

The model can also be investigated to further describe the stages of action, and the dynamics of dialogue. DM interfaces are typically rich in functionality and bear a variety of interactive techniques. The model can be used to investigate links between dialogue techniques and user responses. The alternative paths of action may bear further decomposition. Also, the nature of the reasoning that users employ may be investigated further. Accounts of user behaviour, such as those offered by Lewis (1988) and Young et al (1990) require examination for their appropriateness and coverage of DM interaction.

The model proposes types of error linked to phases of the model. These are an important component of the model as they seek to classify error types. In particular they refer to system-led errors. The error types serve as a taxonomy of system-led dialogue breakdowns. Empirical studies can demonstrate whether the coverage provided by this taxonomy is accurate. Also, the appropriateness of the 'boundaries' of the error types may be examined. A problem with taxonomies is that they may not be orthogonal. Empirical evidence may be used to assess whether the error types have the right level of granularity. Whether they account for all the useful distinctions between error types may also be investigated.

3.8. Chapter Summary

In this chapter a model of DM interaction has been described along with its theoretical motivation from the literature. The model provides a description of interaction in a display-led environment. User memory is accounted for as distributed between user and system, with the system image providing cues to memory, as well as indications of novel feature utility. The cycle of DM is described for both known and novel actions. A description of alternative responses to errors is also provided.

The next chapter describes an empirical investigation of the model. This investigation validates the premises of the model, and also provides further examination of the description of critical interaction phases. The study will also investigate further the nature of errors, error responses, and learning on DM systems.

Chapter 4 - A Study of Interactive Behaviour and Errors by Novice Users of MacDraw 1

4.1. Introduction

This Chapter empirically tests claims made in the model of action. It reports the study of a typical group of novice users on a DM interface. User behaviour patterns are studied, along with error behaviour and user verbal reporting of reasoning and interface interpretation. The study had three broad objectives. These were:

• The validation of the basic model of action concepts: The model contains several possible interaction paths, and behaviour patterns associated with errors. It also assumes that the interaction relies more on environmentally-cued action than on planned task-action sequences. Therefore, evidence was needed that the behaviours described in the model reflect the nature of interaction. This was studied quantatively by observing and categorising user behaviour. In particular, evidence for the opportunistic nature of user action should be a relative lack of planning, and considerable searching and feature exploration. Evidence against the model would be a high frequency of deterministic planning.

• Investigation of Model Granularity Through Study of Users' Problems: Users verbalise problems with reference to expectations, search strategies and interpretation of screen information. These are used to consider the coverage provided by the error categories described in Chapter 3, and in turn, the coverage of critical points in the cycle by the model.

• A deeper analysis of the influence of both signals given by the interface, and users' task-models held in and activated from long-term memory: The subjects were asked to verbalise their reasoning for decisions. This was analysed for evidence of the influences behind behaviours leading to errors. In particular, the knowledge sources that influenced user behaviour were recorded and analysed. Where critical incidents were located, the users' models of tasks were compared to the system model expressed in the actual design. This was done to elicit mismatches between system command sequences and users' expectations.

4.2 Methods

4.2.1. The MacDraw I Package

Empirical data on user interaction with MacDraw™ were collected. A screen dump of the main screen is shown in (Figure 4.1). MacDraw I was chosen because it is a relatively 'pure' direct manipulation package, relying on the use of metaphor, visual cues and feedback. The basic task activities involved are straightforward writing and drawing tasks. The manipulations involved reduce to four basic operations, namely select, drag, drop and type. These four basic manipulations cover a rich variety of functionality. The options for creating objects are visible in an iconic array (the palette) on the left of the draw space. The menu bar contains a number of commands for operating both on selected objects and on the draw space. Also, shapes and lines can be directly modified using select drag and drop options. Therefore, the range of possible dialogue types and dialogue breakdowns is sufficiently broad to test the validity of the model, and investigate possible modifications.

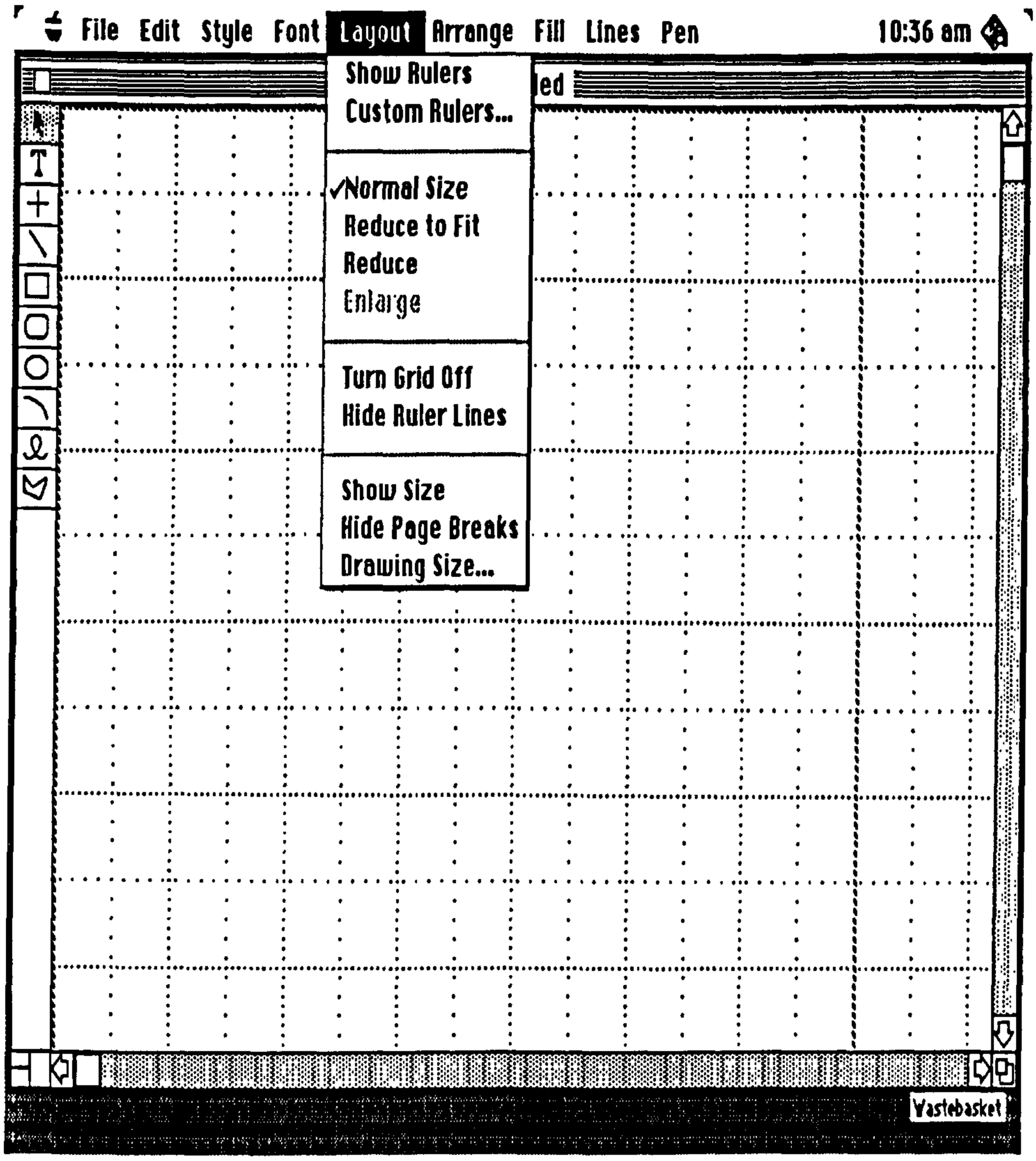


Figure 4.1. The MacDraw I Interface,With a Pull-down Menu Displayed

Another motivation behind the choice of MacDraw I was that it afforded the opportunity to perform a follow-up study investigating the effect of design iterations in a later version. The next version of the package (MacDraw II) was available, and contained some interesting contrasts from its predecessor in functionality and presentation. This study is reported in Section 4.6.

4.2.2. Subject Group

Eight MSc students (6 male, 2 female) took part, all of whom had substantial computer experience. Four of the subjects had never used windows or graphics before, while four (C,D,E,F) had some experience of DM-graphics and window systems. One of the latter group (F) had experience with graphics packages and MacDraw (which he had used twice). The subjects were considered suitable as they were all competent computer users who had no substantial experience with Draw packages. This enabled analysis of the effect of external world knowledge on task performance.

4.2.3. Prior Training

Subjects were given a brief explanation of the basic functions, and then had 10 minutes to explore the system. This ensured that they had familiarity with the basic manipulations and high-level operational principles of the package. After this they were presented with a paper sketch of a data flow diagram and asked to reproduce it in a 30 minute session.

4.2.4. Scenario Design

The scenario approach of presenting a sketch to be reproduced was preferred to a comprehensive list of written instructions. The motivation for this was to allow as much leeway as possible for subjects to order sub-tasks and select their own approach. The scenario contained circles, square boxes, three-sided boxes straight and curved lines with arrows, and text. Each item was present on the diagram at least twice, affording observation of learning, memory of feature use, and possible repeated errors. The scenario is shown in (Figure 4.2.) .

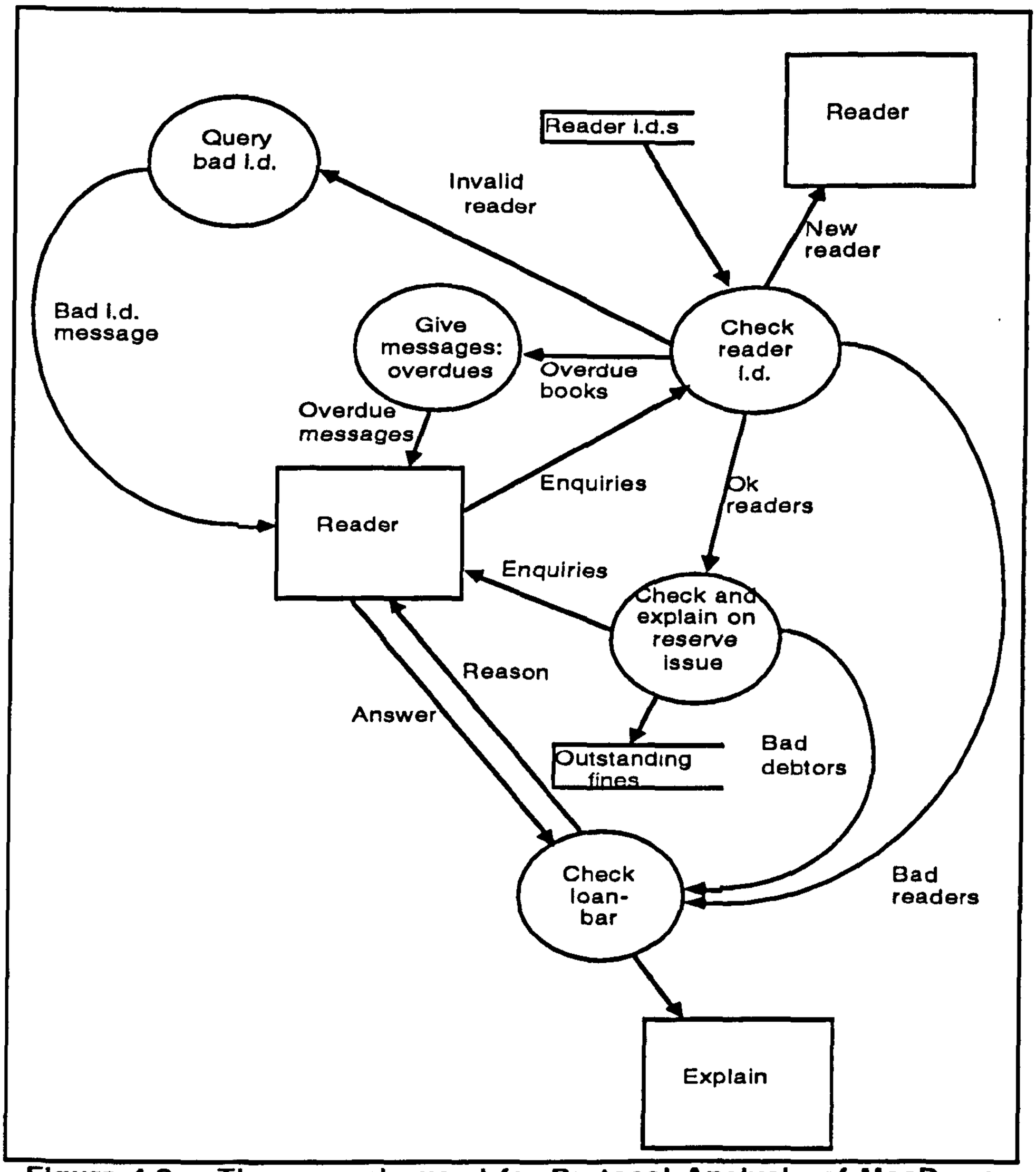


Figure 4.2. The scenario used for Protocol Analysis of MacDraw

4.2.5. Concurrent Verbal Protocols

Subjects were asked to provide verbal protocols on their thoughts and actions, following the procedures of Ericsson and Simon (1984). Sessions were video recorded and user verbalisations were subsequently transcribed. Errors and problems with interaction were noted by the observer during the session. On completion of the experiment users were questioned about the problems that they had experienced and their model of the system, and asked to explain their problems when 'breakdowns' in interaction had occurred. The procedure was similar to the York Manual approach (Wright and Monk 1989) except that questioning by the observer was reserved until the end of the evaluation session because we did not want to disrupt the flow of interaction and we were interested in the compounding effects of errors which may have been alleviated by observer intervention.

4.3. Data Analysis

4.3.1 Interaction Model Categories

The data were analysed to give a summary of interaction for each subject. The following activity categories were used to describe steps in each session. The categories were motivated by the model described in Chapter 3. The following list describes the categories detected from the protocol data.

• Task Action - Any user action that results in constructive performance of the task, and is not declared as exploratory by the user.

• Planning - Where subjects expressed a task strategy, e.g. intention statements beyond the next observed action.

• Feature Search - Scanning of the screen by the subjects possibly accompanied by verbal statements of search for cues.

• Feature Explore - User action declared as trying out a system feature, either for task-action or repair action.

• Repair Action - Action to rectify errors, where the user has a strong belief about what is required (i.e. not exploratory). Repairs may not be preceded by error diagnosis if the reason for an error is immediately apparent.

• Error Diagnosis - Verbalisation of reasoning about an error and associated action. Users must specifically express that an error has been made.

The categories correspond to stages in the model of action, and in turn, to Theory of Action stages proposed by Norman (1986). The Search/explore categories were added, drawing on concepts in the theory of exploratory learning proposed by Lewis (1988).

4.3.2. Model-based Error Categories

The error categories linked to nodes in the model of action, were used to categorise user problems. Protocols were analysed in combination with the observer's notes of users' problems and errors. Problems which resulted in errors (breakdowns in the Wright et al (1989) definition) were placed into six categories (the five described in the model, along with the extra category 'slip' described below). The initial category definitions were validated by a pilot protocol involving two MSc students at City University who had no previous Macintosh experience. The six types of error were:

1. Slips

Minor errors which could be ascribed to failure in attention rather than mistaken intentionality by the user, following the distinction made by Reason (1986). Slips were lexical -keystroke errors which were usually immediately corrected by the user.

2. Misleading Cue:

When an interface feature or state change in a feature (e.g. icons, messages and menus) gives misleading information resulting in the user making an incorrect inference about the effect of a system action or the nature of a system state.

3. Expectancy of an action that is not possible:

When users attempt to find an action which is not supported by the interface. This type of error is typically a result of a model mismatch between user and system. An action is present in the user's model of the task but absent in the system model.

4. Hidden Functionality:

In this error condition, the system does provide the functionality desired by the user; however, the cues provided by the system are insufficient to enable the user's discovery of the functionality. Alternatively, the user may discover hidden, but undesired functionality by accident or random exploration.

5. Inappropriate functionality:
In some cases the system's operation makes sense to the user, but the feature does not quite do what the user requires. The user has to make some compromise and accept degraded functionality.

6. Missing/Ambiguous feedback:
Feedback after an action is either absent or ambiguous, thereby leading the user into making erroneous inferences about the system state and effect of actions.

The categories were used as heuristic classifications, to help define the nature of the errors made in the sessions.

4.3.3. Goal-Tree Analysis

Along with the direct categorisation of errors, further analysis was conducted using a variation on the goal-tree analysis used by Keiras and Polson (1985). Model-based analysis of errors started with analysis of each breakdown supplemented with data from retrospective questions about the incident. The procedures of CCT for constructing user and system models are elaborate and time consuming. As we wished to produce an economical method, a simplified version for deriving the user's model from questions about goals and the user's expression of anticipated actions was adopted. The subjects were asked to repeat the task manually (i.e. without the computer) and the sequence of operations was noted. A preliminary goal tree was constructed from this analysis and then validated by asking subjects to 'walk through' the model and confirm that it was a typical representation of their intentionality. Individual subjects' models were aggregated to form a canonical model. The system model was constructed from analysis of the interface command structure by asking an expert user to complete the task with MacDraw and noting the sequence of actions. As with the User model a preliminary goal tree was constructed and then validated by an independent expert who was asked to assess the reasonableness of the interpretation. User and system models were then interpreted for each breakdown which had been classified as an error.

4.4 Results

4.4.1. Analysis of Users' Activity

The first analysis was a 'time line' view of each session using the activity categories. The resulting diagrams illustrate the subjects' activity which was composed of task actions interleaved with error diagnosis and repair and, less frequently, planning and feature search/exploration. Two sessions are shown in (Figures 4.3 and 4.4.). Figure 4.4. includes annotations denoting the accidental discovery of a feature (D) and the exiting of a task due to the subject's inability to work out the procedure (E). The sessions exhibited considerable inter-individual differences, however some common trends were apparent.

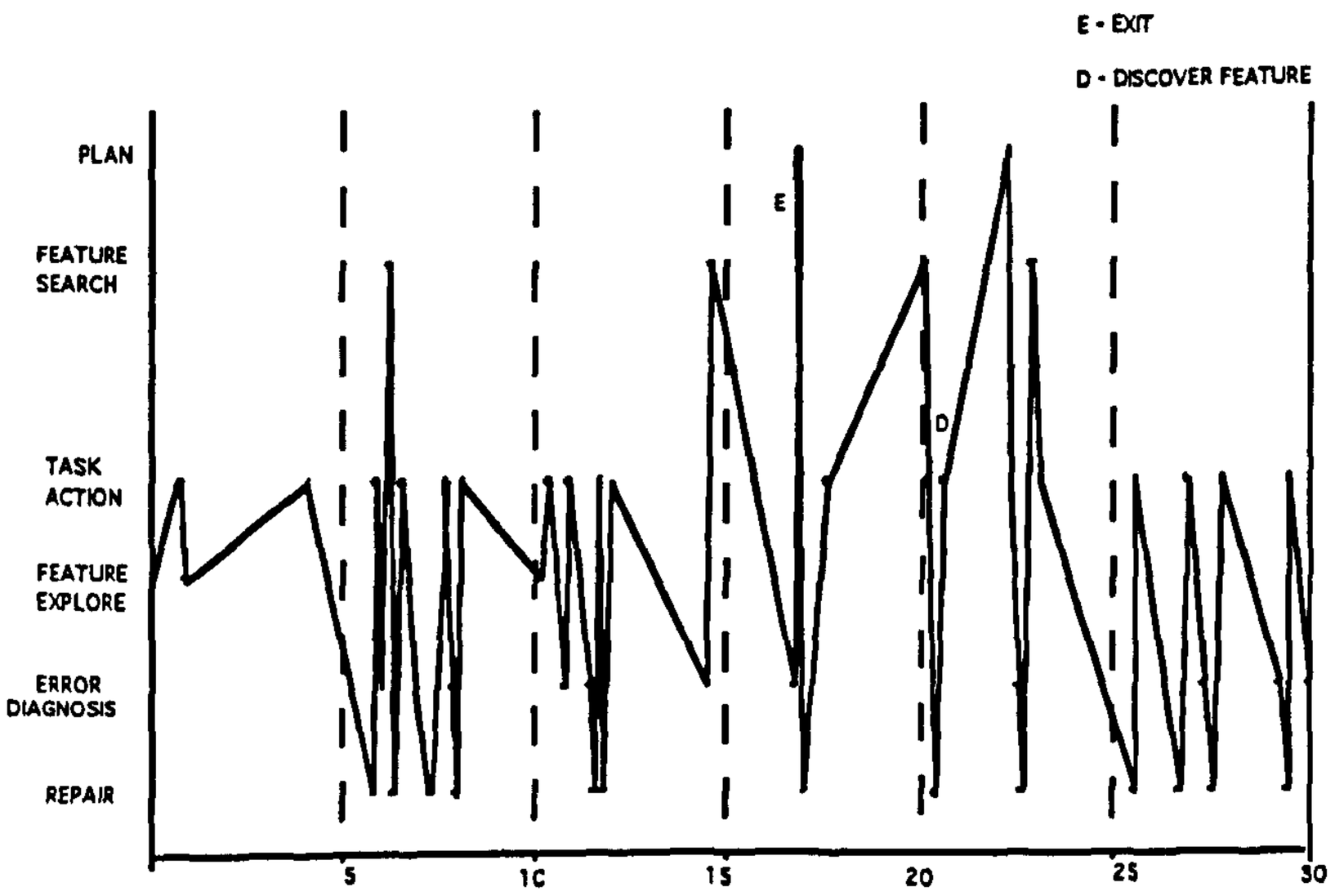


Figure 4.3 TIME-LINE GRAPH ACTIVITY FOR SUBJECT B

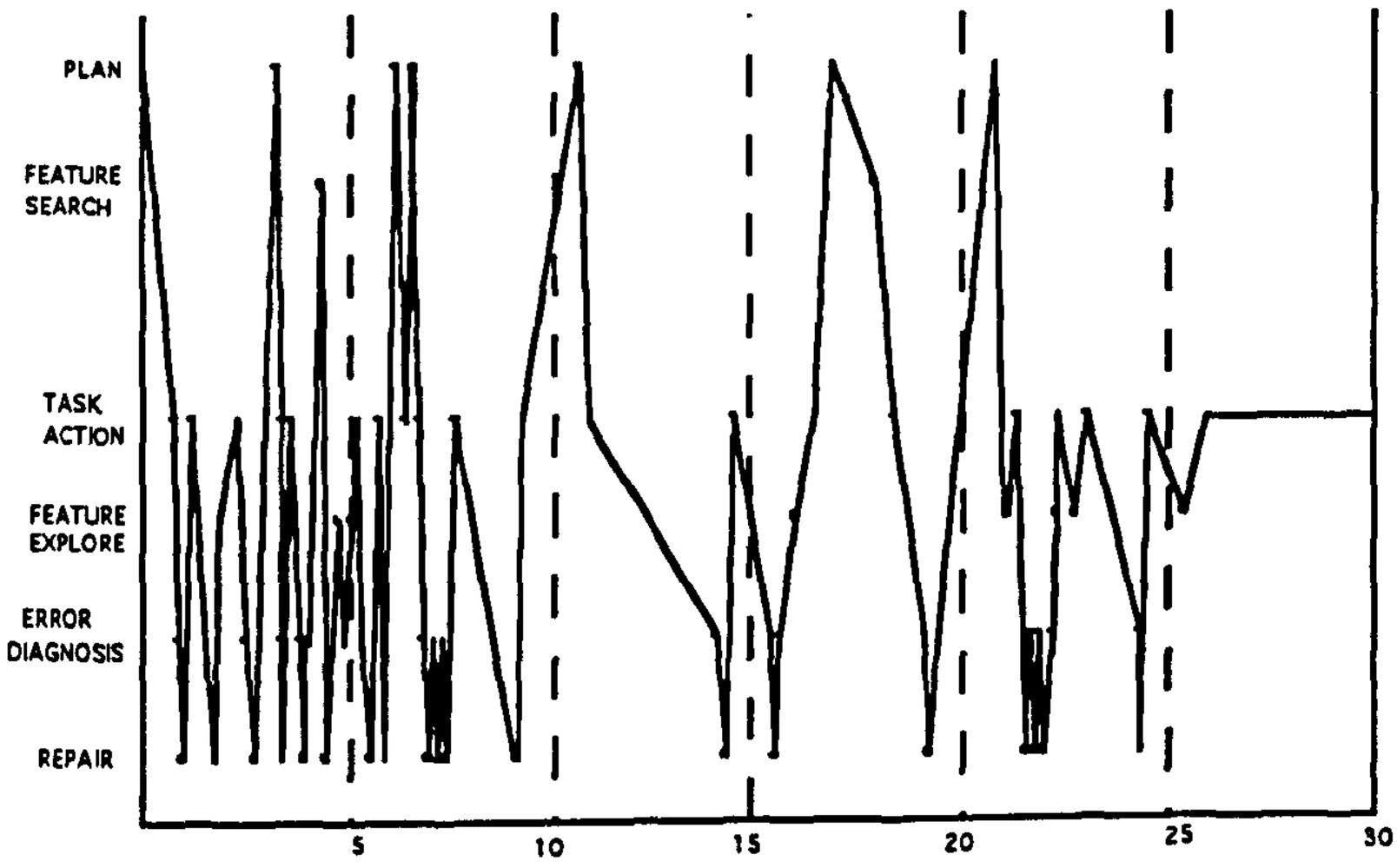


Figure 4.4 TIME-LIME GRAPH OF ACTIVITY FOR SUBJECT C

Task -action was punctuated by errors and exploratory action throughout all sessions. The most common pattern was task action, error diagnosis and repair. Planning, feature search and feature explore were infrequent; however two individuals (D and E) did show considerable feature exploration during the first 10 minutes. No correlation was apparent between subjects' experience and error rates. Subject C (fig 4.4) planned regularly throughout the task. Task-action was followed by error diagnosis and repair early in the sequence although errors decreased as the session progressed. Subject B (fig 4.3) shows less planning than subject C, possibly indicative of a more event driven approach. This individual searched and explored features, however, the predominant pattern was task action alternating with errors. This pattern continued throughout the session.

Individual differences in prior knowledge may be reflected in the patterns. For example, Subject C who had used other drawing packages spent most of the first eight minutes of his session in an error/repair cycle, and then had a relatively trouble free session. This may have been caused by previous knowledge interfering with system operation. Once the new device model had been acquired interaction became relatively error free.

To summarise activity patterns, transitions between the categories were analysed by counting the frequencies Task Action was followed by Error Diagnosis, Repairs, etc. The resulting frequencies were converted into a network diagram to illustrate the pattern for all subjects. The network (see fig 4.5) shows frequent interactions between the Task-action, Error diagnosis and Repair activities. Task-action is central to the main cycle of interaction and error correction. Feature Explore is associated with Repair and Error diagnosis less frequently, suggesting an alternative trial and error style of interaction. Feature search is linked to task-action so discovery of new features appears to occur in normal action rather than in error correction activity. The overall pattern agrees with the interactive sequence predicted by the model,and furthermore the analysis shows which sequences are more frequent than others.

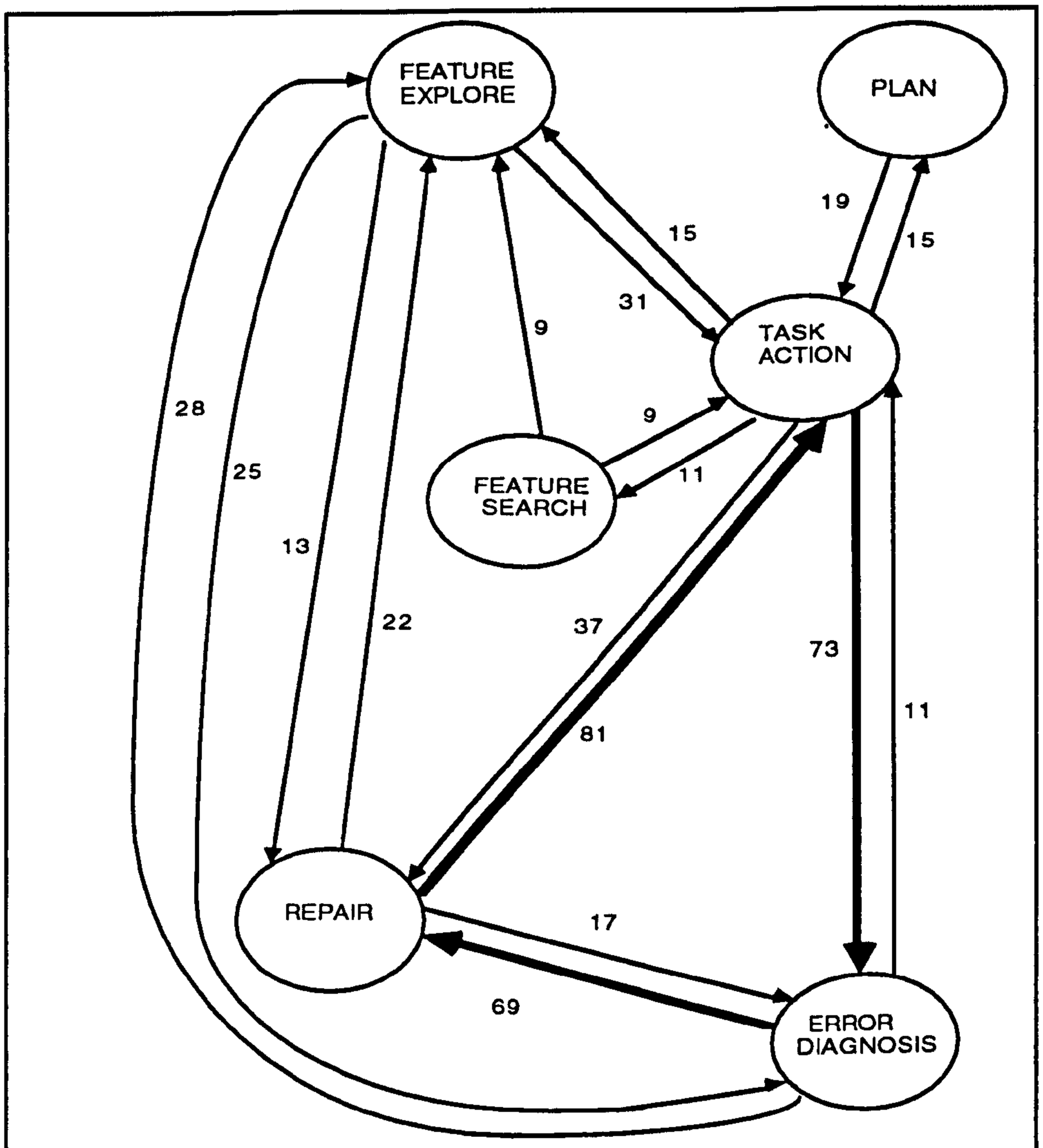


Figure 4.5 Network diagram illustrating pattern of activity for all subjects. Frequencies are binary transitions between activity categories. Only values > 5% total have been shown.

The termination of a task-action sequence was associated with the subjects' perception that they could not work out the needed procedure or when they found that the system functionality was inadequate. One example is the freehand draw. Part of the task was to connect two circles with a curved line. All subjects decided against using this option when they saw its limitations, i.e. the fact that it is impossible to draw a neat line.

Twenty five occurrences of Feature search were recorded. Seventy-two per cent of searches were not associated with errors. Of these, the subject's search was directed on 13 occasions, i.e. their action or verbalisations suggested a target for their search. All these searches attained their objectives. The 6 other searches had no obvious target and only one was successful. Error connected searches (28%) were all non directed with a 50% success rate. In 12 cases the subjects searched menus for cues without knowing what feature they were searching for. Of these, 6 were in direct response to errors. The other 6 were cases when subjects had expressed a new goal and desired task functionality was not apparent.

Feature explore was observed 75 times. Most feature exploration followed discovery of errors (55%). Other Feature explorations were simple checks of functionality (35%) and testing to extend knowledge about a feature's functionality. On six occasions subjects either checked what a feature would do in a specific situation, or tried out new actions. Accidental discovery of new features only occurred three times. Subject A, while attempting to duplicate using the Copy menu option, noticed Duplicate in the same menu during Feature Explore. The same subject discovered the lasso function after an error in trying to move objects, while Subject B pressed the Escape key in a repair action and found that it deleted selected objects.

4.4.2. Exploration and Feature Discovery

The incidence of transitions to feature search from normal task-action was relatively low. This was despite the fact that the scenario required the use of at least three and possibly more palette options. This may partly be explained by the visual salience of the palette which allowed the user to scan so swiftly that search was not observed or verbalised. Another contributory factor may have been that subjects had quickly become familiar with the concept of the palette very early in the session. The nine examples of feature explore leading to task action were all cases of users looking for options at the lower end of the palette. This region of the palette contains variations on the basic shape and line drawing facilities. They declared that they were not sure about the presence of the features, but confident of the appropriate operations once the

features had been found.

Feature discovery through accidents or errors only occurred once. On this occasion the subject forgot to select an option before dragging and accidentally lassoed a circle and text. Four other subjects dragged the lasso by accident, therefore seeing the 'hand and rope' cue, but did not express any understanding.

4.4.3. Model Coverage of User Behaviour

The Chapter 3 model predicted that deterministic planning would not be a significant influence on user action. Of nineteen declarations of planning, four were made right at the beginning of sessions. Subject C declared

'I know what its like drawing circles where you can't fit the text, so i'll do the contents of the circles before I draw them'.

Subject C used knowledge of pen and paper drawing to decide on task ordering. Subjects A and G declared that they would draw all the shapes prior to lines and text, as they felt that placing objects appropriately would be difficult. Other examples of planning tended to come at clear break points in the task, such as the point where all shape-drawing was complete.

Some frequently observed user behaviour had not been explicitly predicted by the model. The model assumes that first time use is linked to guessing of action ('no familiar feature' therefore 'guess and try action'). However, there were a number of cases where the use of a novel feature involved actions expressed as familiar by subjects (i.e. feature search --> task action) . The following extract is taken from Subject A's session. Subject A wishes to ensure all the drawn circles are the same size as one he has already drawn.

Physical - Goes to Layout menu, scans. To Arrange menu. Scans.

Mental - *I'm trying to copy to ensure they're the same size. Mind you, I can see another way of doing it.*

Physical - Goes to Edit
menu.....

Mental - *It seems to me that if you go to the copy part of the menu.... this is what I call copying ---------- I've suddenly realised there's a*

duplicate as well. I'll see what it does.

Physical - Goes to circle, selects. Goes to Edit menu, selects Duplicate (new circle appears).

Whilst Subject A had used a novel feature, he was confident about the operation of the feature. Examples of strong user expectations about the operation of novel features was also frequent in use of novel palette options. This suggests that the link between novel feature use and 'guess action' is not a necessary one, as suggested in the model. Similarly, some account should be made of exploratory action on known features. This also is not accounted for in the model. For example, subjects who knew the operation for moving objects attempted to modify it for a rotation of an object.

A distinction can be made between types of user response to errors. Some subjects responded to errors by selecting alternative means of achieving an action. This course of action is not explicitly referred to by the model. For example, two subjects who were unable to use the arc and freehand facilities opted to use the diagonal line feature to draw 'curved' lines.

4.4.4. Summary of Model Analysis

The proposed model of interaction was supported by empirical data, although explicit system exploration was infrequent. This was despite the fact that a number of novel features were used. This suggests that exploratory learning cannot be characterised as simply piecemeal discovery and learning of features. High-level learning of metaphor concepts seems to create strong expectations about the presence of certain features within the system, and about procedures for their use. This would explain why a large amount of novel feature use was not reported as exploratory by the subjects.

The basic premise, that action is opportunistic rather than the result of deterministic planning, seems to be borne out by the lack of declared plans. All subjects expressed a heuristic strategy for ordering the task early in the sessions. Typical heuristic strategies were to put shapes before text and lines last (5 subjects). This was cited by those subjects as the best way ensure that the objects were properly laid out. There were another eleven cases of planning. However none of these eleven declarations involved device specific declarations of action. The evidence was that user plans amount to general task orderings as opposed to precise procedures. More specific device operations must, therefore, be event or context driven. There is little to suggest that users' knowledge of the task space generates detailed expectations about device operations.

The model based analysis demonstrates a number of cases where exploratory and normal task action result in errors being diagnosed and repair action being performed. However, the nature of those errors, and in turn the adequacy of the level of granularity expressed in the model requires further examination. The model attempts to flag critical points and behaviours in the cycle of action. To have the desired level of explanatory power, those critical points must be linked to the points of interest for evaluators, namely types of interaction breakdown. Therefore, the need for addition and modification to nodes in the model will be conditioned by analysis of errors linked to those nodes. If the recorded errors within categories are diverse, a division of the model into extra stages will become necessary to account for this diversity.

4.5. Analysis of User Errors

4.5.1. Frequency of Errors Within Categories

A total of 132 errors were observed, 28 % of which were "slips', i.e. minor errors which were immediately corrected. Slips were caused by failure of attention and motor-perceptual coordination (e.g. keystroke errors, mouse button problems) and as such could not be attributed to design features of MacDraw software. Model based analysis of slips was superfluous as the causality of the problem was immediately apparent to the user and observer. The other 95 errors were counted as mistakes in which some interpretation of a problem was necessary.

The most frequent error-types were misleading cues (23%), impossible actions (27%) and missing/ambiguous feedback (22%) -see Table 4.1. The distribution of errors among the subjects was reasonably even, apart from subject A who accounted for 48% of the hidden functionality errors. These errors were associated either with attempts to draw arrow-heads on lines or with problems in using the grid-alignment feature. Other high error scores, from subject H (impossible actions) and subject F (misleading cues), were also associated with the arrows feature. Inappropriate functionality was the least frequent error-type being shown by four subjects with the free-hand draw, arcs and flip/rotate functions. Although errors were reasonably evenly distributed among the subjects and by error-type, the association with design features showed considerable differences, as can be seen in Table 4.2.

	A	B	C	D	E	F	G	H	TOTAL
Misleading cues	2	2	-	4	7	2	4	1	22
Expectancy of an action that is not possible	3	2	3	2	2	5	1	8	26
Hidden functionality	9	2	-	1	1	-	2	6	21
Inappropriate functionality	-	-	-	3	1	3	2	-	9
Missing/ ambiguous feedback	2	3	3	4	1	2	1	1	17
Total	16	9	6	14	12	12	10	16	95

Subjects with some experience

Table 4.1. Frequency of error-types by subject

Arrows	47	18 (MC)[7]	16(IA)[7]	13(HF)[2]
Palette Selection	12	7 (FB)[4]	5(IA)[2]	
Drawn Objects	10	5 (HF)[5]	3(FB)[2]	2(IA)[2]
Text	6	3 (IA)[3]	3(FB)[3]	
Freehand draw	4	4 (IF)[3]		
Cursor	3	2 (FB)[2]	1(HF)	
Arc	2	2 (IF)[2]		
Flip operator	2	2 (IF)[1]		
Grid	2	1 (HF)	1(FB)	
Layered Window	2	1 (MC)	1(IF)	
Polygon	1	1 (FB)		
Palette Default	1	1 (MC)		
Copy	1	1 (MC)		
Duplicate	1	1 (MC)		
Rotate	1	1 (IF)		
Total 95				

MC Misleading Cue
IA Expectancy of an Impossible Action
HF Hidden Functionality
IF Inappropriate functionality
FB Missing of ambiguous feedback

[-] No. of subjects who made error

Table 4.2. Frequency of errors ranked by design feature and error category

The arrows function accounted for 49% of all errors, followed by palette selection and misinterpretation of objects' state. In more detail the errors by category and design feature were as follows:

Misleading cues

The arrows feature accounted for 82% of errors in this category by misleading users into expecting that the arrow selection denoted left or right arrow direction (see Figure 4.6), when in fact direction was determined by the start or end point of a drawn line. Seven out of the eight subjects experienced this error and four subjects had repeated errors with this feature.

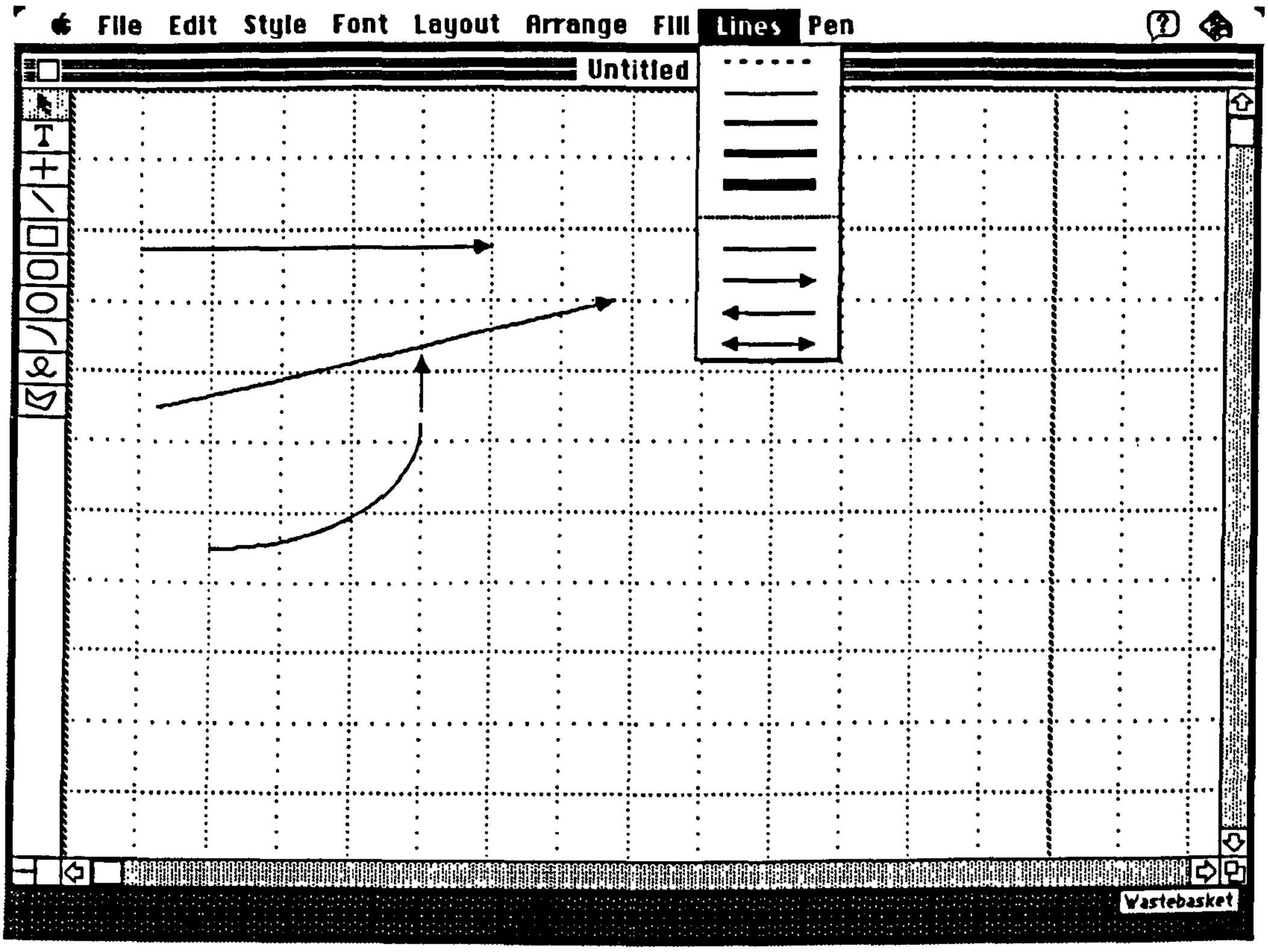


Figure 4.6. MacDraw 1 Screen, showing the arrows menu and arrows on curved arc problem

Impossible actions

A variation on the arrows problem caused 50% of these errors in which users tried to draw an arrow on a curved arc or polygon line, an action not supported by the system.

All subjects encountered this problem and one persisted in trying to solve it without success. Other errors in this class were attempts to repeat drawing the same object when reselection from the palette was necessary (20%), and attempts to place text diagonally.

Hidden functionality

Most errors in this category (62%) also implicated the arrows feature. Users selected the arrows option but did not realise that this set a default for line drawing. When they subsequently attempted to draw lines, considerable confusion was caused by the unexpected appearance of an arrow.

Inappropriate Functionality

These errors were infrequent and did not cause a complete breakdown in interaction as users completed the action, although they were dissatisfied with the result. Subject F reported a problem with the flip and rotate functions which did not produce the type of rotation required. Three subjects verbalised problems with the freehand draw function which produced irregular lines unless the mouse was used carefully.

Missing/Ambiguous feedback

There were two main causes of the 17 feedback related errors. A first problem was inadequate feedback on object selection, accounting for 41% of errors in this category, all of which involved the palette selector highlight. The errors occurred when subjects selected a palette option (causing the icon to highlight) but did not immediately release the mouse button which resulted in the previous option remaining selected. The option highlighting appears to be insufficient to make users aware of the error. The first problem occurred when three subjects who had written text in a circle reported errors when it appeared to obscure the circle's edge and a related problem when on three occasions, subjects dragged circles over written text and were mystified when the text disappeared.

4.5.2. Errors, Repair and User Behaviour

All slips, i.e. simple failures of attention or motor coordination (see Reason 1986), were successfully repaired immediately or within a short time of being recognised by the user. The system feedback was adequate in the case of slips (e.g. selecting the wrong palette object) for the user to quickly interpret the problem. Of the remaining 95 mistakes, only five were successfully repaired with the subjects verbally demonstrating understanding of their causation (4 Feedback, 1 Hidden functionality). Ten errors were fatal in the sense that part of the task had to be abandoned and work thrown away. Abandoned actions were caused by one hidden functionality error

(lasso), four inappropriate functionality problems (freehand draw, curved arc) and five impossible actions when subjects attempted to place arrows on arcs and rotate text. In the remaining errors (76%) subjects continued the task sub-optimally with another operation.

Subjects often hypothesised that failed actions implied a system constraint, when, in fact, the action was possible. For instance, two subjects who placed a circle over text (thus blanking the text) remarked that 'it won't let me to do it' and 'it looks like you can't have it'. In some cases errors were compounded, for example Subject H, while unsuccessfully trying to place an arrow on an arc, inadvertently set the arrow-line default. Subsequently he encountered problems with unexpected arrows when drawing lines.

On 25% occasions when a problem was encountered, the subjects verbalised a hypothesis about what went wrong and tested their hypotheses by system exploration. Problems were solved infrequently (5 times); one example was subject C who tried unsuccessfully to place an arrowhead on a very small line. He reasoned that the line was simply to small to bear an arrow, experimented with a slightly longer line, and was proved correct. In most cases, however, the subjects' hypotheses were unsuccessful and provided no explanation.

After a minority of errors (18%) no strong explanatory hypothesis was verbalised and the subjects took experimental action with several features. They engaged in apparently random interaction or sequential search, exemplified by Subject C who went through several menu options looking for an operation to place text in front of a circle. After the majority of errors (57%) the subjects verbalised an explanation for the problem but did not appear to attempt to solve it either by reasoning or guesswork. Overall, complete error recovery was poor even though the subjects showed some understanding of the error causality. Most preferred to try alternative actions to complete the task.

4.5.3. Model-based Analysis of Error Causality

To understand why errors occurred in more depth, a separate analysis was conducted, based on the goal-tree analysis of Keiras and Polson (1985). System and user models of interaction were compared for the more frequent errors and for errors which were reported by several subjects. It is important to note that the system models represented the most typical action path for a task. However, an important attribute of DM systems is the flexibility of interaction and many different system models are possible. Indeed many errors were circumvented by the user following another action path.

4.5.3.1.Arrows on lines

Use of the arrows feature caused problems in two situations, selection of the direction of arrows and placing an arrow on a curved line. The user and system models for the arrows feature are illustrated in Figure 4.7. There is a considerable discrepancy between the two models which is accounted for by selecting the arrowhead direction before drawing the line. The user's model (for novice users) has no selection step as arrows are drawn as a feature at the end of a line. Problems caused by not selecting arrows were recoverable, although some users took some time to learn that they had to select the arrows option then draw the line.

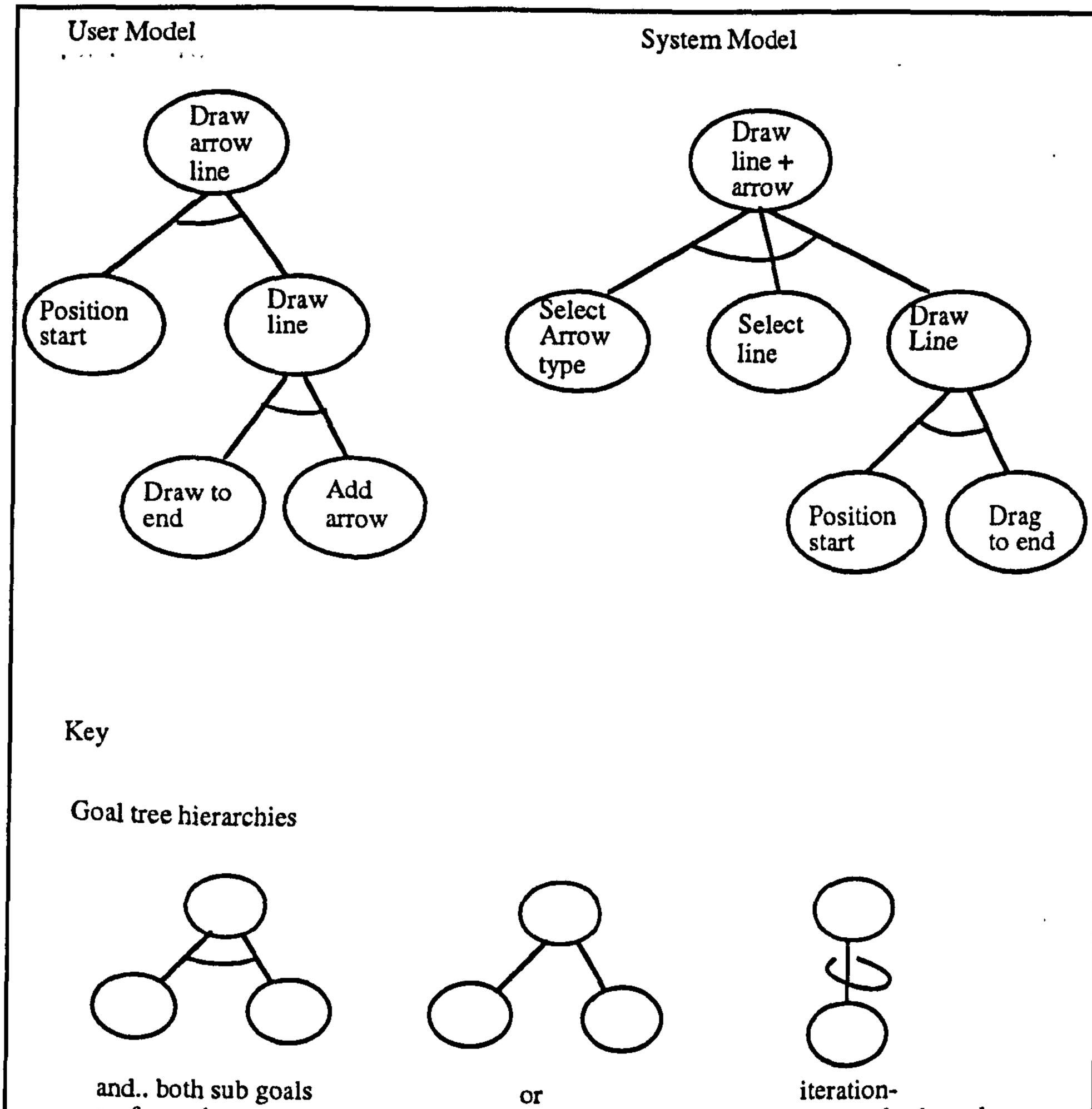


Figure 4.7. User and System models for drawing a line with an arrow using MacDraw I

4.5.3.2. Arrows on curved lines

The user and system models show even more disparity in this case, as illustrated in figure 4.8. The system imposes an artificial restriction that arrows can only be drawn on straight lines. Hence curved lines (and other palette options such as the polygon) can not be selected with the arrows as a sub-option. Recovery from this error was difficult for nearly all users and impossible for two, who never discovered the solution of drawing a small straight line bearing an arrow on the end of a curved arc. This error was classified as missing functionality and is illustrated clearly by model mismatch analysis. However, this error also involved visual aspects of the interface. Users naturally expected to be able to draw arrows on a curved line, and stated that their expectation was reinforced by the system model which did allow arrows on straight lines. Users attempted to draw arrows on curved lines by selecting an arrow option then selecting the curved line (or less frequently the polygon) in the belief that allocation of the arrows attribute would be implemented by the system. In MacDraw II this design problem is still present and in follow-up experiments the same errors were observed.

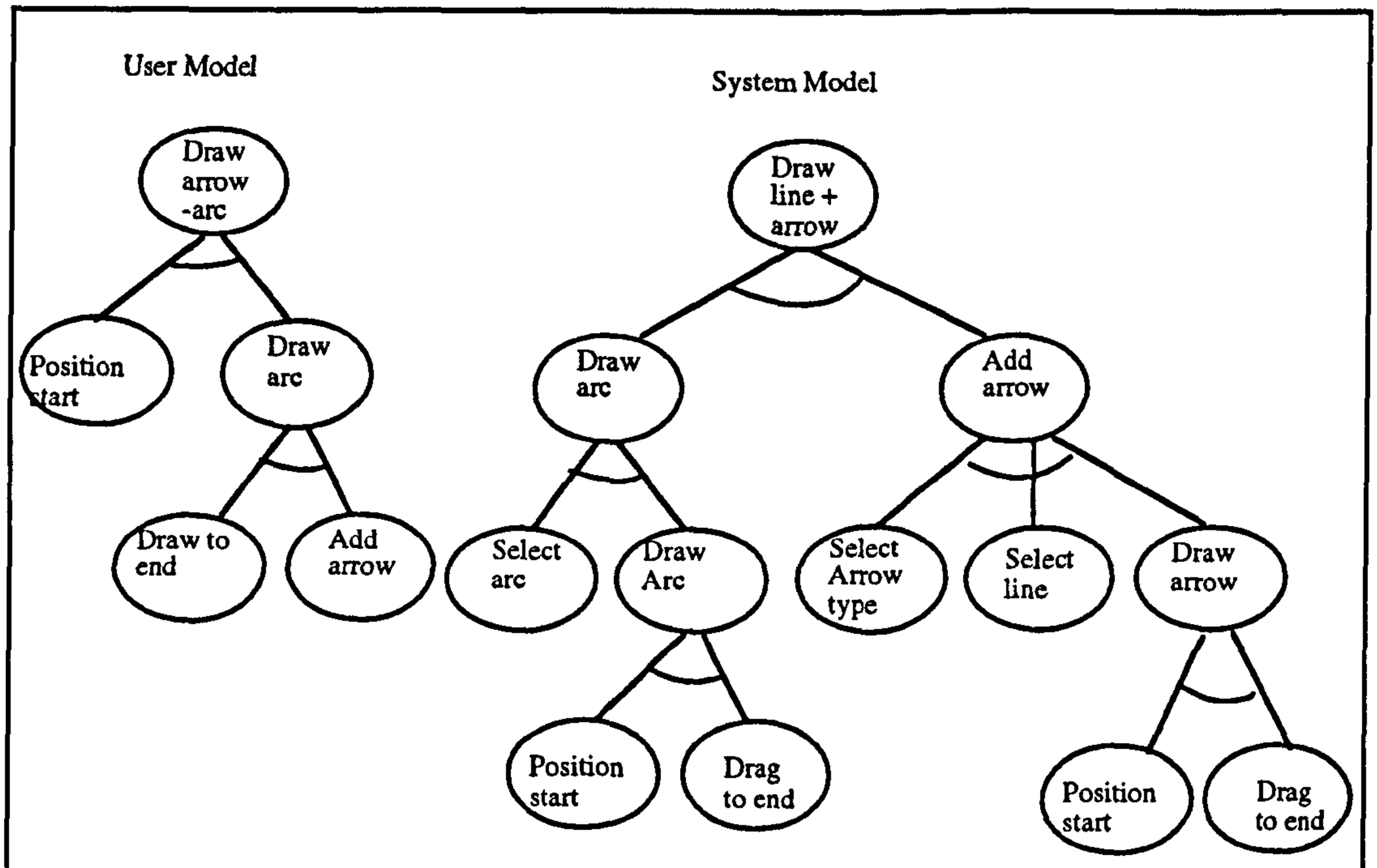


Figure 4.8. User and System models for drawing an arc with an arrow using MacDraw I

4.5.3.3. Autopositioning shapes: The Grid feature.

The grid feature automatically positions shapes and lines accordingly to a set of system coordinates rather than the absolute position shown by the cursor. The grid option also influences moving shapes which 'jump' in small steps between preset coordinates rather than moving incrementally under direct cursor control. Even though the Grid option is explicit in the Layout menu, this feature gave users problems in a variety of contexts. The user - system model clash is caused by an additional action of autopositioning by the system (see figure 4.9). The problem was manifest when subjects wanted precise positioning of objects. The system confounded their attempts by 'jumping' object positions during move operations. This error was classified as hidden functionality as the system performed an action which was neither expected by the user nor overtly cued. Understanding the whole problem involved cues as well as actions. The successful solution to the problem required that the subjects find and then

correctly interpret the grid option on the layout menu. However, the cue 'Turn Grid off' did not trigger further investigation.

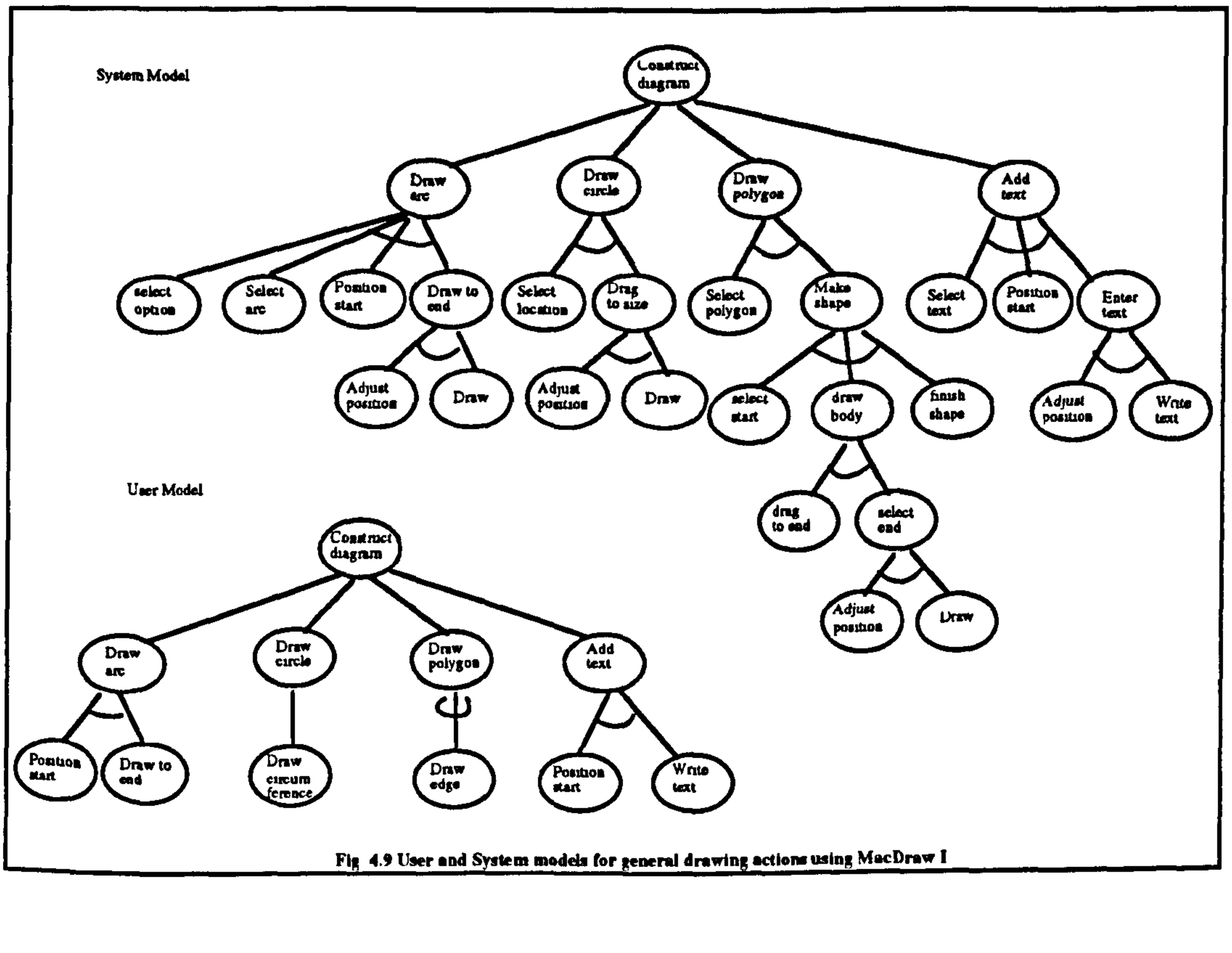


Fig 4.9 User and System models for general drawing actions using MacDraw I

4.5.3.4. General Object Creation

The system models in figure 4.10 illustrate another frequently reported user problem with selection and defaults (critical incidents), although these did not always lead to errors (complete breakdown). The system model dictates that the user has to reselect the palette operation each time a new shape is drawn. Furthermore, feedback for option selection was inadequate leading to many errors when users were unaware of the system state. This frustrated several users who expected their first selection to become a default, allowing them to use the same object several times. In this case an

interesting design dilemma is posed. If a system model assumed this default then it could clash with a user model in which single instances of many different shapes were to be drawn, and hence an explicit select is required. An optimal design solution may not be possible, as this feature illustrates the problem of a design model having to satisfy multiple user models.

4.5.3.5. Moving overlapping shapes

These problems were not caused by any model mismatch even though the user and system models do not show a complete correspondence (see figure 4.11a). Instead the problem is entirely a matter of feedback. When users drag a shape over another one, their expectation is that the first (moved) shape will still be visible inside or underneath the second. The feedback during the dragging suggests that the shape is transparent (the text and ruler lines are visible underneath). However, on releasing the mouse-button the shape becomes 'solid', obscuring the object underneath.

Another example of this was selection of multiple objects (see figure 4.11b). Subjects tried unsuccessfully to select several objects by serial points, unaware that a group objects (lasso) facility existed. Some subjects saw the hand-icon cue and didn't understand it. Others never saw the cue as it only became visible by holding the mouse button down on unoccupied drawing space and then moving the cursor with mouse-button down to open the lasso.

4.5.3.6. Summary of Mismatch Analysis

The analysis suggest that there may be two levels of design problem that lead to user errors. The first is clearly a problem of the cognitive design of task-action sequences. The influence of users' task models on expectations dictates that steps in the device model must, wherever possible, link to equivalent steps in the user's model. However, the degree to which this is possible is likely to vary from task to task. It is inevitable that the structure of some tasks on the device will not resemble domain-task structures. For example, the concept of a plotting restriction or grid is unlikely to figure in a typical user's domain model. Mismatch analysis demonstrates that the success of this depends on the presentation of task steps.

A number of the errors reported in the study could not be traced to task-design as such, but rather to the presentation of those steps. In other words, some errors can only be adequately explained with reference to visual aspects of the design. In a number of cases the designer (or whoever is responsible for making improvements) has to judge whether the optimal design change involves altering a step, or altering a presentational aspect of the interface. The 'Grid' problem shows this potential

dilemma. It is arguable that the 'structure' of interaction can beneficially be altered by having the autogrid as an option rather than a default. It may also be argued that if the cue was altered from 'Turn Grid Off' to something more comprehensible, usability problems may also be avoided. The study of MacDraw II in Section 4.6., below, provides further practical examples of these issues.

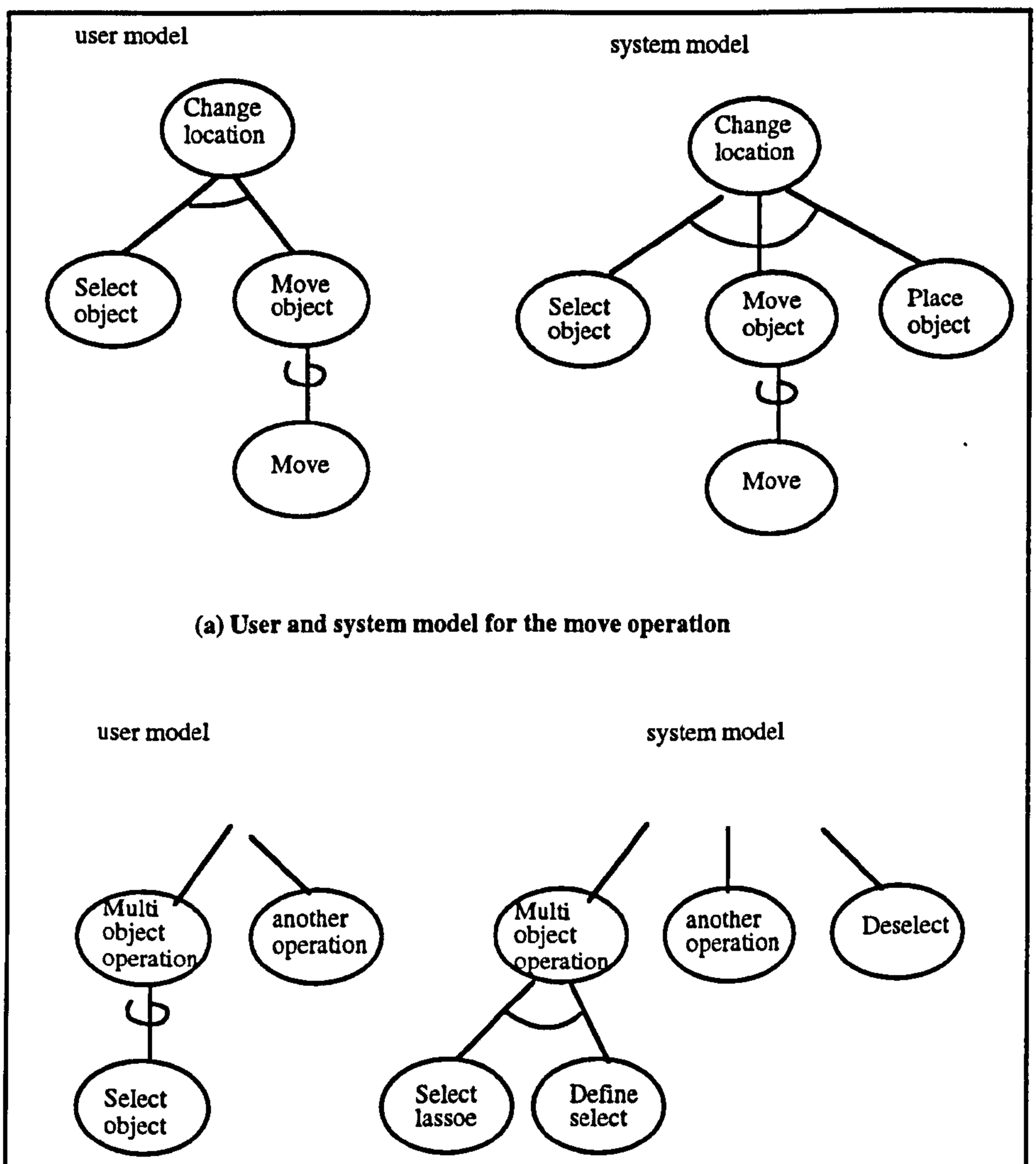
4.5.4. Further Analysis of Error Classifications

The categories used for describing errors were analysed to investigate possible sub-categories. Individual examples of categorised errors were re-examined for common characteristics. It was found that most categories could be sub-divided. The following list gives examples of the errors assigned to those categories.

4.5.4.1. Misleading Cue Errors

Examples in this category could be divided into two classes. One involves the user simply *linking a feature to the wrong task.* The examples of this error type showed cues which caused incorrect assumptions about their relevance to tasks. For example, Subject H reasoned that the cursor default at the top of the palette was 'the arrow section' and was part of the arrow selection procedure for lines. The cursor default was intended to (and does) resemble the on-screen cursor. However it also resembles the arrow menu cue and arrowed line-ends (Figure 4.7. shows the palette with the cursor default shaded).

A second type of cue problem is where the user makes a correct assumption in linking the feature with the type of task, but is misled about the *specific effects* of the feature. The 'arrow menu' example, reported above, comes into this category. The menu is iconic, showing a line with an arrow on the right, and one on the left. The subjects who made this error were misled by the assumption that directionality was indicated by the cue.



(a) User and system model for the move operation

(b) Partial User and system models for the multiple object (lassoe) operation

Figure 4.10

4.5.4.2. Expectancy/Impossible action errors:
The examples that were found can be broadly split into two types, namely task/device mismatches and unsupported assumptions from previous actions. In the former cases subjects were compromised by device constraints. For example, there were 5 cases of subjects attempting to draw a second shape without reselecting the necessary palette option. The subjects involved complained that the system was doing something that they hadn't asked it to do. The real-world user model involves no equivalent reselection action.

The 16 instances of subjects attempting to add an arrow to an arc line, an action that the system does not allow, demonstrating the second type. Subjects' comments strongly suggested that they had reapplied knowledge of other arrow utilities. The arc is grouped on the same palette as the line options for which the arrow facility can be used, and is functionally similar in all other aspects.

4.5.4.3. Hidden Functionality Errors
One type of hidden functionality is exemplified by the grid problem reported in Section 4.5.3. This is the problem of cues not being recognised by the user. When two subjects were asked retrospectively to identify the feature they both guessed that it would remove the graph lines that were visible on the screen. Neither could connect the cue name to the correct functionality.

Another example of this was the selection of multiple objects. Others never saw the cue as it only became visible by accident (i.e. when the cursor is dragged with no option or object selected). Crucially, this was compounded by the fact that the subjects did not expect such functionality to be activated in that way. Three subjects admitted that they would look in the menu for such utilities. This seems to present a separate problem from simply cueing functionality. Users also require cues for the selection of search locations.

The other type of hidden functionality error contrast with the previous two examples. This was where a feature was accidentally activated without the subject being given any indication of what s/he had done. The 13 errors (between 2 subjects) associated with arrow default settings were an example of this. The subjects performed what they saw as straightforward arrow selection procedures, not realising that they were setting a default. The only indication that the default has been set comes on a subsequent occasion when a line is drawn. Subjects are left to deduce from memory how the default came to be set in such a way. The only two subjects who stumbled

upon this problem could not work it out even after experimenting with the feature. This type of problem is described as a mode error by Monk (1986).

4.5.4.4. Inappropriate Functionality Errors

These errors can be divided into two categories. One is the inexact matching between cue and task. For example, Subject F had a problem with the flip and rotate functions. He admitted that the type of rotation he had in mind was different from the available effect. The features in question all indicated their function to an extent, but left users to guess about their precise effects.

The second example is simply where the demands on user manipulation skills are too great. For example, all subjects who tried to use the freehand draw option conceded that it was too awkward to use accurately.

4.5.4.5. Missing/Ambiguous feedback errors

We identified 3 elements that made up the 17 feedback related errors. The first is the straightforward failure to adhere to the WYSIWYG principle. An example of this occurred when 3 subjects who had written text in a circle reported errors when it appeared to blank out the circle's edge.

A second problem was a discrepancy between continuous feedback in mid-action, and post-action feedback. The best examples of this were the 3 occasions when subjects dragged circles over written text. The feedback during the dragging suggests that the shape is hollow (the text and ruler lines are visible underneath). However, on releasing the mouse-button the shape becomes 'solid', obscuring the object underneath.

A third problem was false confirmation, the illusion that an action has been successfully completed, accounting for 8 of the 17 errors in this category. Seven of these involved the palette selector highlight. The errors occurred when subjects selected a palette option (causing the highlight to move to that option) but did not immediately release the mouse button. This caused the previously selected option to remain current, with the highlight returning to that option. Although the selected option is shaded, this feedback seems to be insufficient to make users aware of the error. Subjects admitted that they had mistakenly believed that the highlight had confirmed a successful reselection.

4.5.5. Summary of Category Analysis

Whilst all mode errors fitted easily into the categories, it is clear that the categories'

diagnostic power is limited. The analysis shows a considerable diversity of examples, suggesting that a more precise set of critical points in the action cycle can usefully be described. For example, a misleading cue may cause incorrect identification of a feature, but also (as in the freehand and arc examples) effect inaccurate expectations about the feature's operation. Also, hidden functionality errors sometimes refer to the problem of locating features. However, in other examples the problem is one of secondary state changes, or hidden mode changes. In these cases the system has changed mode resulting in known user actions producing different effects when subsequently used, undermining example-based learning. The examples of expectation of impossible action errors also suggest a range of mismatch types which designers may usefully investigate to find the optimal alterations to a design.

4.6. Studies of Expert Users

4.6.1. Methods

A follow-up study was conducted using two highly experienced subjects (Subjects J and K). Both subjects had used MacDraw regularly for between two and three years. Subject J had constructed a number of data-flow diagrams, similar to the scenario task. Subject K had not drawn diagrams in the particular style of the scenario, but had experience of similar drawing tasks. The experimental conditions (time, scenario, data analysis) were the same as for the novice subjects.

4.6.2. Motivation

The motivation for studying experts was to identify fundamental contrasts between novice and expert users. The Model of Action describes action across the spectrum of user types. Therefore, expert user behaviour should (if the model is accurate) resemble the description of behaviour described in the model.

4.6.3. Results

Subject J completed the task in fourteen minutes, and Subject K completed it in 19 minutes. Both Subjects J and K spent most of the session in task-action, with only occasional feature search (Subject K scanning for arc and datastore drawing options). They were able to perform long sequences of task action, adjusting the sequence of operations to suit specific needs. For example, Subject J was able to anticipate at each stage whether the 'grid' should be turned on or off. Subject K used the 'Duplicate' facility extensively, creating all the required shapes prior to arranging them. Subject J

did not report any feature search. Subject K only found it necessary to search for arc drawing facilities.

The two subjects made a total of three errors between them. Subject J experienced difficulties plotting with accuracy having forgotten to select the feature 'turn grid on'. Subject K moved a composite object (a three-line 'datastore' and text) expecting it to move as one. He also experienced problems using the arc feature (he claimed not to be familiar with the feature).

4.6.4. Summary of Expert Study

The marked absence of search actions suggests the use of compiled knowledge. This is also evident in the long sequences of unbroken task-action, and the anticipation of steps. In particular, both subjects were able to anticipate and go to features which were not immediately visible (e.g. menu options). This is fundamentally different to the Chapter 3 model, which explicitly states that all action includes a search phase.

4.7. Studies of MacDraw II

4.7.1. Motivation

We investigated MacDraw II to contrast aspects of the interface design with MacDraw I. There were three reasons for looking at this. The first reason was to see which problematic parts of the interface design had been changed. The second was to assess the effectiveness of those changes, given the flaws that had been uncovered. The third was to look at the effect of other functionality that had also been added.

4.7.2. Methods

The investigation involved inspection of the MacDraw II interface and a pilot evaluation with two subjects. The inspection involved a simple scan of functionality, listing the changes that had been made, and whether these corresponded to features that caused problems in the original study.

The pilot evaluation involved the same procedure and scenario as for the MacDraw I study, in order to direct users to the same task-action behaviour. The sessions were recorded and analysed for use of specific features, and error behaviour, rather than for a full analysis of user behaviour.

4.7.3. Results

Twenty one design problems had been identified with MacDraw I. Of these 12 remained unaltered in the updated package and only 4 problems had clearly been rectified.

4.7.3.1. Altered Design Features

The menu for arrow selection uses linguistic rather than iconic cues to provide a clear description of the functionality. The palette is now selected as soon as the user places the cursor on an option, removing the ambiguous feedback present in the previous version. The text feature has no ghost border to obscure other shapes.

In three other design features there is some doubt as to whether the changes will solve usability problems e.g.:

Arc- The user is given more (mid-action) visual information about the angle being plotted. Lines appear on the screen showing the X and Y co-ordinates of the arc. However, the user is given no extra help in anticipating or controlling automatic repositioning.

Arrow default Selection- The arrow selections are denoted by a marker next to the selected option. However, this feedback is not visible when the menu disappears from view.

Grid menu- The 'Turn Grid Off' menu option has been renamed 'Turn Autogrid Off'. The name refers to a system function which may not be immediately apparent to the user.

4.7.3.2. Other New Features in MacDraw II

Some design features which did not create problems for our MacDraw I users have been changed in MacDraw II, without delivering any apparent improvement. The on-screen zoom facility is denoted by a mountain-like icon, a 'mini metaphor' which neither subject managed to discover despite both expressing the need to look at the screen in finer detail.

The grid in MacDraw II causes even more exaggerated repositioning than MacDraw I. Subjects complained that placement was particularly difficult.

4.6.4. Summary of MacDraw II Study

The evidence from the study suggests that there are considerable problems in satisfactorily improving interface features even after problems are reported. The fact that a feature is identified as troublesome is simply a step in the process. Further investigation of the true nature of usability problems is necessary to avoid wrong or inadequate alterations. Some of the changes seemed to cause greater problems than before, suggesting the need for assessing the effects of changes to design features in advance of new versions being produced.

4.7. Conclusions

In this chapter we have looked both at the validity of the model-based approach, and at the possibility of refining the descriptions within it. Further to this we have considered how model-based techniques can make explicit the real nature of errors in terms of the user's strategies and reasoning, and the interface's influence on that reasoning.

The model of Action's basic description seems sound when used to describe patterns of interaction observed in the sessions. However, the model's utility for evaluation can only be fully exploited if it more explicitly describes critical interaction points.

While the error categories gave some preliminary guidance towards error causality, model comparison was required for more complete understanding. Whilst the data suggests that the proposed categories comfortably contain the user errors that would be of concern to evaluators and designers, their diagnostic power is limited. The weakness of the categories used is that they fail, in some cases, to provide a sufficiently accurate level of diagnosis, which pinpoints the root cause of a problem. Therefore a further analysis of the critical points in interaction is necessary, in order to account for the diversity of interaction problems, and yield recommendations for design improvements.

However, mismatch between models in terms of user or system actions is only part of the story. In direct manipulation interfaces the perceptual aspects of interaction and the role of metaphors have to be taken into account. Users' problems need to be considered from different viewpoints:

(i) Users' problems which may result in an error or just degraded functionality; attributed to a mismatch of actions between the system model and user's model.

(ii) Users are unable to predict machine operation from cues, metaphors and feedback, even though they may understand the system model.

(iii) The potential for error/problem recovery afforded by the system; and users' ability to interpret the system state from feedback and metaphor and then find appropriate, remedial actions.

Errors may be minimised by task analysis and user modelling to ensure that actions are designed in the system model which correspond with users' expectations. Thorough task analysis may help specification of actions, but successful design in DM systems also requires careful attention to metaphors, cue and feedback, all issues implicated in the detail of interface design. Mismatch analysis in this context requires analysis of how task model nodes are represented. This is required as the interface has to effect mappings to a range of task elements including action object, goal state and the enabling state for the next action. Some mismatches, such as confusion over feature identity from cues, may be described as failure of interface items to effect meaningful mappings to task model elements. Helping users' understanding of system operation by metaphor and explicit cues appears to be a vital component of successful design, although current state of the art guidelines (e.g. Smith and Mosier 1986, Browne 1988, Laurel 1990) can only provide a partial guidance.

4.8. Chapter Summary

This chapter has tested claims made by the model of action, through empirical study of user behaviour, and errors. The model provides a broad description of user behaviour. However, descriptions of expert behaviour and the effect of the display on novice user behaviour require more detailed descriptions. Also, there is scope for a more detailed description of error types. The following Chapter proposes a modified model of interaction accounting for user knowledge levels as well as sources of user knowledge. A further taxonomy of errors is also proposed.

Chapter 5 - Further Model of Action Developments

5.1. Introduction

This Chapter develops the model of the action cycle, and proposes an approach for its use in evaluation. The model described in Chapter 3 is split into three sections, describing three levels of mental processing by users, using definitions proposed by Rasmussen (1993). These levels are defined by the knowledge-spaces that are recruited from by users. Examples are provided demonstrating that search and operation specifying mental acts are distinct steps in the action cycle, and may require different types of system support. Error types observed in the Chapter 4 study are linked to stages in the models. These are described as system failures to fulfil roles in the dialogue (i.e. support for specific user mental acts). The latter section of the chapter describes ways in which dialogue role failures may be pinpointed using data from user-based studies.

5.2. Revisions to the Model of Action

5.2.1. Introduction

The following sections describe the elaboration of the Model of Action in the light of the Chapter 4 study. Examples from the studies show the need for models of DM action which can distinguish between levels of expertise. The model of action shown in Figure 3.1. assumes that search, trying a feature, and evaluation, provides a generic description of user action. It describes paths of action where a known feature is found, or where the user has to guess the utility of a novel feature. However, the Chapter 4 study suggests that a more precise account of action is required, as mental activity will be determined by the user's knowledge of the task and device.

5.2.2. Expert Users

The two expert user sessions described in Chapter 4 showed a relative lack of feature search. Most interactive sequences were performed automatically by the experts, using compiled knowledge of the system. The reports from both expert subjects suggested that a detailed 'search for candidate features' was not required for most

actions. For example, Subject J was able to declare in advance whether the grid would or would not be required for various phases of the task. The selection of point sizes for various items was also achieved without prior experimentation. This contrasts with the trial and error approach of the novice subjects.

The expert subjects, whilst tracking the system response continuously, rarely paused during task sequences. This suggests that they use compiled procedures, dispensing with the need to closely monitor every state-change. The experts' rich models of the system made them less dependent on feedback as a source of information for subsequent action. For example, Subject K grouped and repeatedly duplicated the first circled text item that he had created. He declared that the text could be rapidly edited after duplication. Novice subjects only used the 'duplicate' feature for shapes, none of which were edited further.

5.2.3. Partial Device and Task Knowledge

Evidence from the study in Chapter 4 suggested that novices users' knowledge was variable in nature. The study showed a strong emphasis on synthesis-based generalisations, as described by Lewis (1988). In Chapter 2, Lewis's view of abstraction was described for generalisation of operational sequences. However, the learning of extended operation sequences using abstraction was not explicitly modelled. The studies showed that MacDraw contains a number of features that operate in a similar but contrasting way. These features are presented in a way which induces abstraction by users (e.g. grouping palette options). The two subjects who tried the polygon feature had both used line and shape drawing facilities. The task facing these users was to infer that extra drag/release and double-click operations were required. Both users were able to work out that the extra drag and release actions were needed. However, both reported errors before learning the double-click action with the polygon.

The sessions suggested that the mapping of task space to device space described by Moran (1983) and Payne (1991) requires a more detailed description to account for partial knowledge and learning. Subjects C and F, for example, had knowledge of the Macintosh (Subject F having used MacDraw), although they still needed to discover much of the system. The package has similar Edit, Font and Style menus to other applications that these subjects had used. The eighteen errors they made were exclusively with features unique to MacDraw (e.g. the Arc), while common features such as 'cut and paste' and other editing facilities were used without error. Subject D used the 'cut and paste' facilities declaring that he knew the feature from a different package. This subject (unlike C and F) had no experience with Macintosh software,

but he was able to port across operational knowledge from the previously used package.

5.2.4. Interaction Level Models

The model of action, described in Chapter 3 was reassessed in the light of the findings described above. This reassessment utilised distinctions drawn by Rasmussen (1986) to describe levels of user action. Rasmussen describes three types of user action; skill-based, rule-based, and knowledge-based. In skill-based interaction the user automatically performs tasks without conscious planning. This is expert behaviour, where device-task procedures have been practised and internalised. In rule-based interaction, the user has fragments of operational knowledge and rules for selecting and applying those operations. In knowledge-based processing the user is calling on external knowledge sources, and reasoning with general, device-independent, knowledge.

Typical DM interaction (for novice and expert users) is likely to be a mixture of all three types. To emphasise how the processing types combine, it is useful to describe the device-space knowledge of an experienced package user. The experienced user will know the majority of device procedures and features well enough for skill-based interaction. However, this user's knowledge of the device-space is unlikely to be complete. When the user finds and experiments with a previously unused feature, rule-based processing will apply. The user will examine the novel feature and select an operation. Also, the user may recognise a similar feature which is known from another system, and retrieve the operation. However, if the system response confounds expectations, the user may be forced to reason from first principles (i.e. knowledge-based processing) rather than from more structured device knowledge. So, even for the highly experienced user, a combination of three types of processing is likely.

The novice user is also likely to use a mixture of the three processing types, but in different proportions to the expert. Some procedures may be internalised rapidly. For example, subjects in the chapter 4 study were asked to create several shapes and move them into a pattern of drawn shapes. It is arguable that, having drawn some shapes, their grasp of the manipulation sequence is sufficiently automatic to be described as skill-based. However, in novel situations, the subjects reasoned about new features to support selection of procedures (explored in greater detail below). As with expert users, novices resort to knowledge-based processing when system feedback is not understood.

Figure 5.1. briefly summarises the knowledge spaces that are utilised in the three types of processing. Knowledge-based processing is used when more specific device knowledge is not available to the user. Rule-based processing may use experience of the current package (e.g. how palette features are operated). The current package may include features which are common across an operating environment with which the user is familiar. Knowledge of packages with a similar look and feel (e.g. knowledge of other WIMP interfaces) may also be used. The user may also apply operational knowledge of a feature specific metaphor (such as cut and paste) from a previously used package. The skill-based user is able to recognise a familiar task and perform a sequence of operations. The processing levels apply both to the device operations and the external task. For example, users of a statistics package (e.g. Macintosh Statworks) will import their skill and experience in operating on statistical data from the external task.

Processing Level	Knowledge Spaces
Knowledge-based	General Knowledge Task domain knowledge
Rule-based	Partial knowledge of current package Knowledge of other device packages Features from other packages Interaction style, look and feel
Skill-based	Automated task-procedure Compiled device/package knowledge

Figure 5.1. Rasmussen's Knowledge Levels with Related Knowledge Spaces

The processing levels proposed by Rasmussen (1986) concentrate solely on how operations are specified. There is no account of how the display influences search and feature selection. Therefore Rasmussen's account only partly embraces the stages of

mental processing described by Norman (1986). Norman's theory refers to searching the environment, and recognising affordances and constraints. Rule-based action may result from search guided by metaphor interpretation or merely guesswork. Metaphor in the large tends to assist knowledge-based processing by suggesting general principles and heuristics influencing search and general operations. However, feature level metaphors are more likely to be action metaphors, stimulating the recruitment of operation rules. Rasmussen does not accommodate metaphors in his model of reasoning, and hence cannot give a complete account of the factors influencing user behaviour at the three levels.

The models to be described distinguish between two types of learning and rules. One is procedural attachment learning, where features of a certain type are bound with operations (if < feature.x > then operation is < A1....An >). In knowledge-based processing the user needs to interpret an action metaphor, deriving rules by mapping between task-space and device space. In rule-based processing, the user will use procedural attachment rules which are already established. These may be rules for known operations already performed (e.g. knowledge of the 'cut and paste' procedure from general computer experience). Also, the user may apply a known rule to operate a novel feature (e.g. < feature recognised = palette option > therefore < apply known palette operation>). There is a distinction between general and specific rules of operation. General rules apply across operations. For example, rules for operating pull-down menus apply to a large number of differing types of menu. Some features, however, will have specific rules of operation. These may be seperate from general rules (e.g. the lasso) or a specialisation of a general rule (e.g. the extra mouse operations needed for the MacDraw polygon palette option).

The second type is locational feature learning which binds display areas to types of features, thereby guiding search (if < recognition criteria > then contains < feature.x >). A user with no prior knowledge of the system must interpret the visual image for familiar features and groupings (knowledge-based search). A user with some experience will have criteria for guiding feature search (rule-based level).

The following sections demonstrate the need to account for the complete process from goal formation, through search to the specification of operations. Similar notations to the Model of Action in figure 3.1. are used. The processing levels are distinguished by the way in which operations are derived in accordance with Rasmussen's account. However, the models also show the influence of multiple knowledge-spaces on user behaviour.

5.2.5. Model of Knowledge-Based Action

Knowledge-based processing relies on task/domain knowledge or general knowledge. The task-domain consists of two relevant types of knowledge. One is knowledge of typical objects and actions, which provides heuristics and principles to guide exploratory action on the device. The other is wider knowledge of object classes and structures in the domain. General knowledge refers to everything which is not necessarily associated with a particular task. Therefore, a general operational metaphor will often prompt use of general knowledge.

A sequential model of knowledge-based action is shown in figure 5.2.a. The user searches the interface for feature cues ('scan interface'). In the absence of known features or groupings, the user must interpret the visual metaphor or scan for cues. The user may find a feature which is visible on-screen or possible feature locations (e.g. menus). The user may be familiar with the domain that is represented by the system. MacDraw, for example, shows a pen and paper metaphor which appears familiar to the user. This may prompt the user to import general knowledge to guide search and feature interpretation.

The user recruits knowledge of how the domain is organised, to inform search of the interface. For example, types of utility or tool may be grouped together in the users model of the task-domain (see Reisner 1990). The user's model of the task-domain consists of classes of objects and criteria linking them together. So, for example, if a kitchen is a domain, the objects within it may be partitioned into groups according to function (e.g. crockery items, cutlery items, cooking utensils). The user may scan, find a feature cluster or group identifier, and focus search in that location.

Some tasks may not be interpretable in terms of a device-independent task-domain. A package may be 'empowering technology' rather than enhancing familiar tasks. In these cases the user has to learn new tasks, rather than translate existing task knowledge. Also, metaphor-based systems are likely to include some functionality which is not interpretable with reference to the task-space. The MacDraw 'autogrid' for example is a device concept with no equivalent in the novice user's task-model. Also, the design may employ a secondary metaphor. For example, a 'human body' metaphor has been used to represent the device domain in visual programming environments (Kilgour 1989). The user may, therefore, need to import knowledge about the secondary metaphor from the task domain.

The model describes the user searching without knowledge of where the feature is likely to be. The user relies on metaphor suggestions either in choice of location

('guess location') or by spotting a salient feature on the screen (e.g. the wastebasket icon). Having selected a feature, the user has the task of selecting the operation ('guess action'). The user may be able to infer the appropriate operation from the cue and general knowledge (e.g. the Wastebasket). In other words, principles may be imported from the metaphor for device operation, as the user is aware that items for disposal generally require carrying to the bin and placing inside. The user may be familiar with a range of basic manipulations (e.g. how to drag and drop) which can be mapped to these general principles. Otherwise, the user has to simply experiment.

The satisfactory completion of the action leads to 'confirm operation' and provides information which may be used in subsequent feature exploration 'add rule'. This transorms part of the overall knowledge-space from the knowledge-based level to the rule-based level. The operation may provide an example from which generalisable system knowledge can be gained. This may be a generalisable rule for applying device operations, similar to that described by Lewis (1988). For example, a user who has performed a font changing task, has knowledge of the operation sequence [select text item - select font]. This knowledge can be recruited in subsequent operations by abstracting the rule [rule = select text - select operation]. Also, knowledge about operations on non-text objects is generated by generalisation [rule = select {_ item} - operation].

The 'font' example above may also yield knowledge which assists in further search tasks. The action may contribute to spatial learning, and knowledge of where utilities are located. The user will have gained some knowledge of the pull-down menus, which will constrain subsequent search. For example, MacDraw I has point changing facilities located in the Font menu. The user may remember this for subsequent interaction. Metaphor interpretation and guesswork (knowledge-based search) is thus replaced by actual system knowledge (rule-based search) as device learning proceeds.

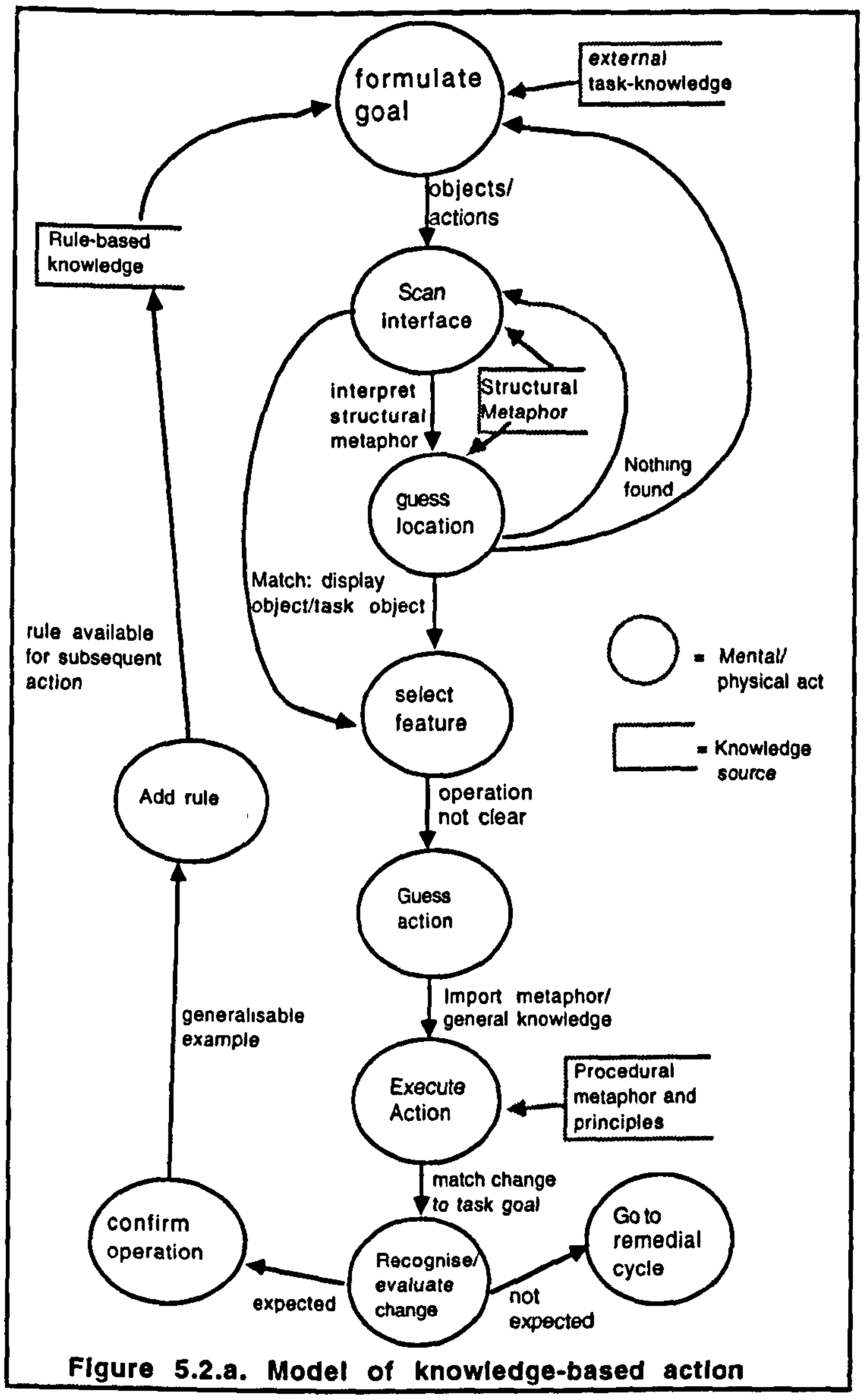


Figure 5.2.a. Model of knowledge-based action

5.2.6. Model of Rule-Based Action

Figure 5.2.b. describes rule-based action. The model describes paths by which rule-based processing may come about. Specifically, it includes all paths of mental action which lead to the direct selection of operational rules. Whereas knowledge-based processing requires the translation of general heuristics of principles to device operation rules, rule-based processing applies known operational rules directly. Some of the paths in the model show knowledge recruitment that is associated with knowledge-based rather than rule-based processing. Specifically, locational rules derived from metaphor interpretation or guesswork, may reveal features which prompt the recruitment of known rules of operation.

The user searches for familiar features or groupings ('scan interface'). The user may have knowledge of particular areas or clusters, with which certain feature types are associated. Therefore the user will be able to apply this knowledge to the current search task. However, it is possible that the user will not be able to select a strong candidate location, having only partial knowledge of the interface. This may prompt the user to use knowledge-based processing, selecting and experimenting with an unknown feature ('go to knowledge-based level'). Also, the user may resort to knowledge-based search, trying to interpret the visual metaphor ('guess location'). This may also lead to the knowledge-based level if no familiar feature is found

Search based on imported knowledge of the task-space may lead to rule-based action, if a familiar feature is found ('find familiar feature'). For example, a user may find familiar 'cut and paste' features during a random menu search. The user may know this feature from previously used packages. If the visual metaphor fails to direct search, the user may try a random search of the interface. The user may find a feature on the screen, during the initial scan. This is represented by the arrow linking 'scan interface' to 'select feature', and marked by 'match feature to task'.

The node 'go to familiar/recognised location' describes two types of knowledge recruitment by the user. One is recognition of an area as being linked to a certain type of feature ('goto familiar location'). The user may, for example, have seen that point size options are located in the MacDraw I Font menu, having entered it for font options. In this case, direct system knowledge is used. In another example, a Word Processor user may have used utilities for customising the interface and setting preferences, and therefore know where such facilities are located. This may help the user in subsequent search for a feature to customising tab settings. The user may abstract the needed facility to a type (i.e. tab setting is a kind of 'system set-up'

utility). This matches with knowledge of where such features tend to be situated, gained from the previously used features.

The alternative type of knowledge recruitment describes search of new areas of the interface. The visual metaphor may act as a cue suggesting the location of certain feature types ('goto recognised location'). The user may direct search knowing neither if a suitable feature is available, nor how such a feature would be operated. In this sense the search is knowledge-based. However, the user may find a feature which bears familiar presentational aspects suggesting operations (e.g. menu type, dialogue box) and apply known operational rules.

The user uses abstracted system knowledge as a rule in applying a known operation to a new feature (retrieve and modify operation). The 'recognise/evaluate change' includes a path back to 'retrieve and modify known operation'. In such cases a general rule has been applied, but an additional rule is needed to complete a particular action. This accounts for features, such as the MacDraw polygon, which may only reveal the need for a modified 'palette operation' as the user performs and monitors the action. Otherwise, a successful action contributes to (generalisable) system knowledge ('confirm operation - add feature').

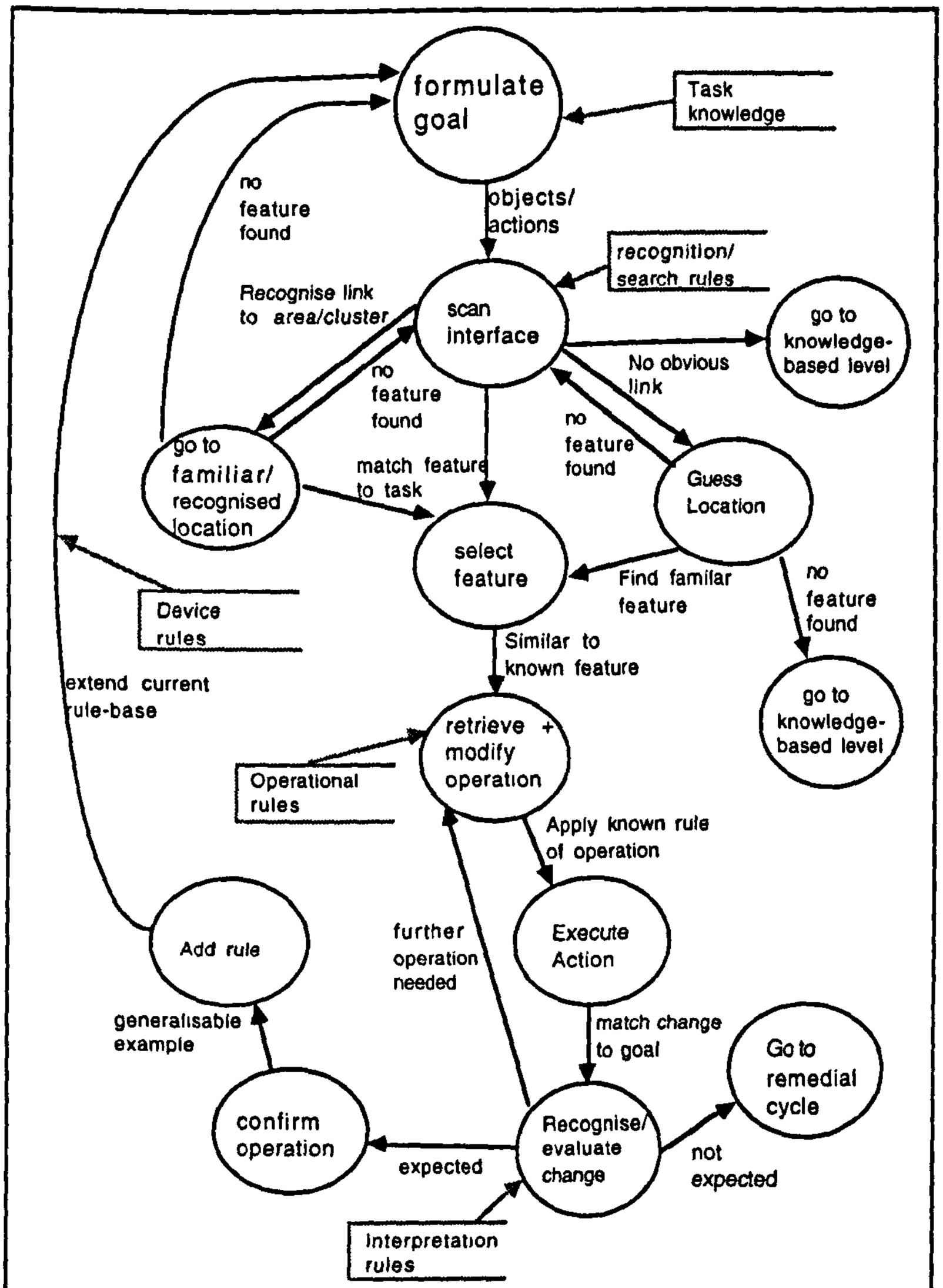


Figure 5.2.b. Model of rule-based action

5.2.7. Model of Skill-Based Action

A sequential model of skill-based processing is shown in Figure 5.2.C. Skill-based action contains relatively few phases, as search and action are automatic. The user selects the necessary operations automatically when the goal is formed. The phases 'goto feature' and 'execute action' are also automatic. This represents 'direct engagement' as described by Hollan et al (1986). The user does not have to devote any working memory or concious processing to the device operation. The phase 'recognise/evaluate change' simply refers to tracking of task progress. The user is not directly checking that the device has been operated correctly, but using compiled knowledge.

Skill-based users automatically recognise the circumstances in which internalised procedures can be used. The system image triggers automatic hand and eye movement ('goto recognised feature') and performance of action. The user goes to 'retrieve operation', with the feature acting as a cue to the knowledge store of operational information.

The 'recognise/evaluate change' node is linked to the 'goto recognised feature'. This describes the way in which skill-based users chunk sequences of familiar actions together in rapid sequence. The 'correct slip' node refers to fact that skill-based action includes the automatic correcting of slips. There is no learning pathway in the skill-based model. Knowledge is already compiled, and no extra knowledge is acquired.

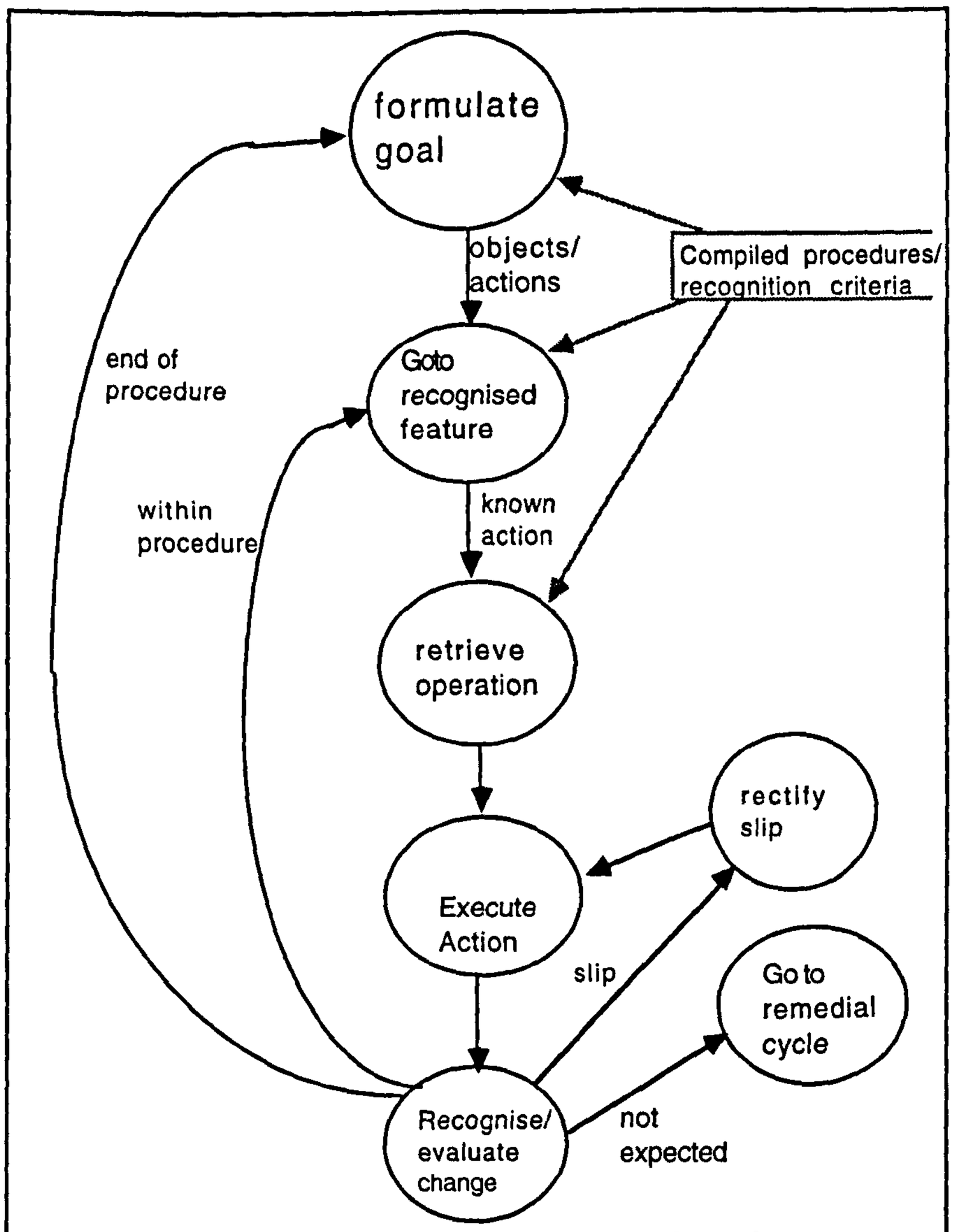


Figure 5.2.c. Model of skill-based action

5.2.8. Model of the Remedial Cycle

This section describes a model of the remedial cycle, which accounts for the effects of errors on processing levels. The remedial cycle expands on the 'error cycle' element of the Figure 3.1. model. The model is shown in figure 5.3. The description of feature discovery through accidents or errors remains unaltered from the original model of action. The studies produced examples of either accidental discovery (e.g. menu options close to the used feature) or manipulation errors producing comprehensible feedback (e.g. Subject A dragging without selecting and finding the lasso feature as a result). In these cases, the features were often immediately useful and prompted the next action, whereas in others, the discovery was internalised for later use. The contrast between errors that contribute to learning, and those which cause dialogue breakdowns is a crucial distinction. The former has been cited as a desirable component of DM interfaces (Shneiderman 1986).

The major modification is an expanded account of users' response to unsatisfactory system responses, particularly where feedback is not fully understood. Unexpected responses are potentially large influences on the development of the user's model of the system. The user will attempt to reason about the output, and this may affect the acquisition and use of rules. The user may either internalise information about novel features and procedures, or review rules that are already held.

The node 'change rule' refers to cases where the feedback prompts the user to use a different operation for the task. The circumstances in which this may happen vary. One situation involves the user trying to find a quick, efficient way of performing the action, already knowing another slower way of performing it. This is an example of the distinction between optimising and satisficing, drawn by Simon (1973). For example, the user may try duplication or group resizing to cut down effort, but can revert to creating individual objects, or resizing if the action fails. Also, the feedback may demonstrate the nature of an error. For example, two subjects failed to double-click after using the polygon option. A further line was produced as they dragged the cursor from the item to the palette, indicating that the system had not confirmed completion. Both responded by performing the correct action.

The node 'modify specification of action' refers to cases where the user believes that attempted action sequence was only partially correct . For example, the user may re-order a sequence of actions (e.g. line draw select, arrow option select, drag line) retaining most of the assumptions that formed the original specification. Also, if a feature does not perform to the expected standard,users may accept degraded

functionality, modify their expectations and redo the action.

The node 'exploratory action/test assumption' refers to action which is not intended as task action, but is intended to test and validate assumptions resulting from analysis of a failed action. The user may attempt to test a feature which had contributed to assumptions about the current action. For example, in the Chapter 4 study, subject H tested the operation for adding arrows to lines by experimenting with the 'diagonal line' option (the arrow feature having failed to work for arcs). He claimed that he expected the arrow option to work for arcs as it had for diagonal lines. This is an example of abstraction-based reasoning (Lewis 1988) proving unsuccessful.

The 'guess feature' node involves the user resorting to knowledge-based rather than rule-based processing. The user does not try to generate a hypothesis about the nature of the problem or the correct action. The user may resort to general knowledge, or simply try action in the hope of discovering something hitherto unknown by trial and error learning.

The node 'add constraint' refers to the effect that a failed action may have on the user's model of the system. The user may interpret the failure of an action as evidence of a system constraint, rather than taking exploratory action. An example of this is provided by the two subjects who obscured text by dragging a new shape over it. As a result, both assumed that it was impossible to place shapes over text items leaving the text visible . In both cases the assumed constraint caused a rethink in the overall ordering of drawing tasks.

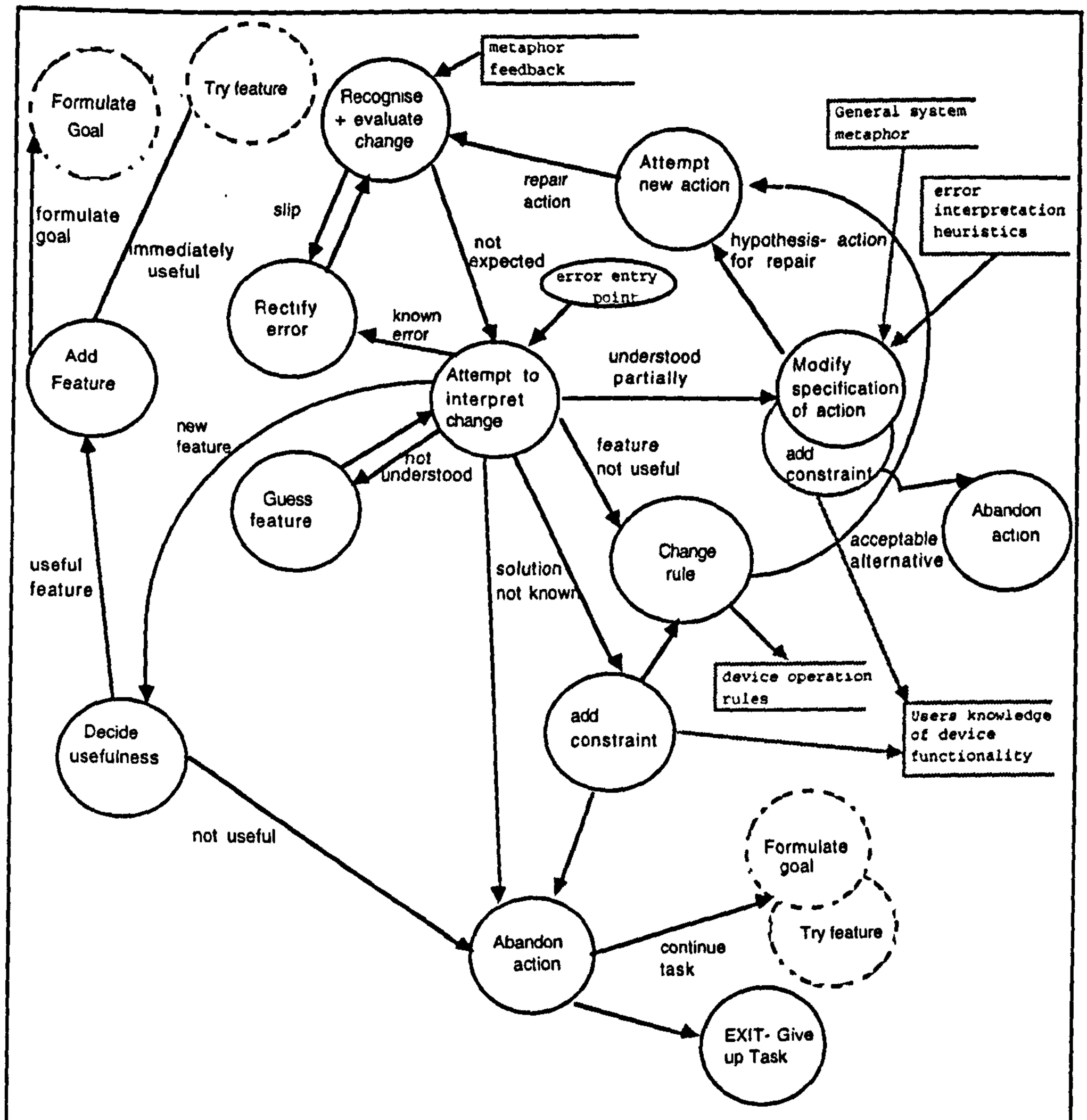


Figure 5.3 Model of the Cycle of Remedial Action

5.2.9. Discussion of Models

The process-level models demonstrate the need to account for the influence of multiple knowledge spaces on user action. Rasmussen (1986) only accounts for the point in the action cycle where operations are specified. This corresponds to the 'try action' and 'guess and try action' nodes from the original model in Figure 3.1. The model of rule-based action has the equivalent node 'retrieve and modify operation'. This describes users either generalising knowledge of the system, or porting across knowledge from the operating system or similar packages and features. The rule-based model has the equivalent node 'retrieve operation'. The phase in the action cycle only partially accounts for the mental activity between the formation of a goal and the execution of an action. A full account must also explain how features are searched for and found. Skill-based action is distinct from other processing levels in that sufficient knowledge of feature location, identity and operation is available to the user at the point when intentions are formed.

The model divides the rule-based level into specific rule sets. The empirical study in Chapter 4 shows that users employ distinct rules for recognition, the specification of operations and the interpretation of feedback. This implies that distinct interface features may support use of each rule set. Also, the model extends the account of processing levels by Rasmussen (1986), by showing that the processing levels interact. Therefore, many user actions cannot simply be described with reference to a single processing level.

The model of the remedial cycle shows how unexpected or unsatisfactory feedback may cause users to reference different knowledge spaces, and use different processing levels. In particular, users may need to use knowledge-based processing after rule-based processing fails.

5.3. Models of Action Specification

5.3.1. Introduction

In the Theory of Action described by Norman (1986), one stage links the forming of intentions to the execution of action, namely the 'specification of action'. The specification stage of Norman's theory is further analysed in this section. This section

decomposes the specification of action, separating search activities and the specification of operations. The importance of separating search and recognition mental acts from the actual specification of operations is described. Broadly, the interface should support each mental activity. Search, feature recognition and operation specification may be supported by separate aspects of the design. In turn, a dialogue breakdown may be traced to a failure to support a particular mental activity and, therefore, a specific element of the design.

5.3.2. Search and the Specification of Operations

The knowledge-spaces described in Figure 5.1 apply not only to the specification of operations, but also to search. It is conceivable that different processing types may be used at different points in the action cycle. Knowledge-based search may be used to find features, followed by rule-based specification of operations. It also conceivable that rule-based search may locate a feature, but the user has to guess or reason about how it is operated. The factors that influence a search may be quite different from those which prompt action once a feature is selected. A model of the knowledge that users may recruit must account for the link between knowledge spaces and specific mental tasks. Search is usefully seen as a separate processing sub-system, distinct from the specification of operations.

Figure 5.4. distinguishes between the utilisation of knowledge-spaces for search tasks and the specification of operations. The 'specify operation' sub-system only describes how the user decides to operate a feature, and not the preceding search. Each knowledge-space (cited in Figure 5.1.) has a contrasting use. Knowledge of the task domain may initially be used to comprehend the spatial metaphor (directing search) or in the recognition of features. Task-domain knowledge may also contribute to specifying operations by facilitating the import (and translation) of operational rules from the task-domain to the device. Similarly, general knowledge may simply inform search (e.g. comprehension of group headers), or choice of operations. Affordances (Norman 1988, Gaver 1991) such as sliders or scroll-bars are designed to provide an indication of possible operations.

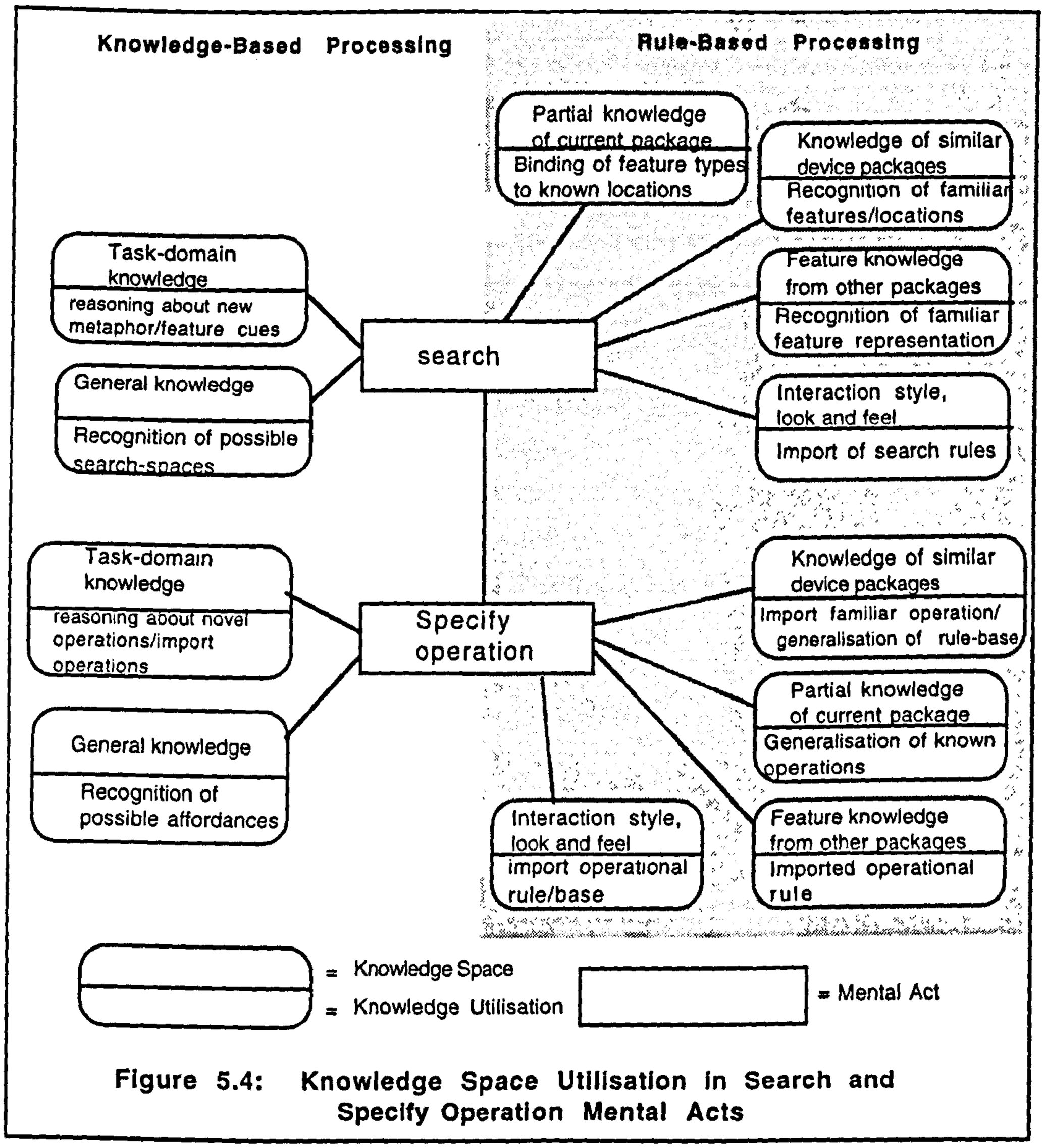


Figure 5.4: Knowledge Space Utilisation in Search and Specify Operation Mental Acts

Knowledge-spaces associated with rule-based processing may also influence either subsystem. Partial knowledge of the current package may include knowledge of where types of utility are located. However, this does not necessarily imply that a feature, if located, will be operable as the user may have to interpret the feature cue (using task-domain or general knowledge). Also, the user may select a search location using experience of the current package, but find a feature cue which resembles a known feature from a previously used package (prompting use of imported rather than abstracted rules).

The following sections contain detailed examples illustrating the influence of different processing levels and knowledge-spaces for particular actions. Four contrasting DM dialogue scenarios are described. These include a brief description of the user's relevant system knowledge, and the current state of interaction. The scenarios describe what the user needs to know, find, recognise, and interpret in each case.

5.3.3. Deletion Using The Wastebasket

The wastebasket example demonstrates the separation of search from specifying operations. Two hypothetical users, with contrasting experience, are described. The two users wish to create disk space by removing some files. User 1 has experience with the operation of dragging and dropping files into directories (but has no experience of the wastebasket). User 2 has no experience of either.

Both users recruit general knowledge in identifying the wastebasket icon. The wastebasket icon suggests a utility for disposing of unwanted items. However, the two users recruit from contrasting knowledge-spaces to specify operations. User 1 is able to recruit abstracted knowledge of how files are moved, having placed files into directories (rule: select file - drag to directory - release button). The rule of operation is used with 'directory' substituted by 'delete buffer'. User 2, by contrast, must translate general operational heuristics or principles from the task-domain to the device domain. The action of throwing an item into the wastebasket maps to dragging and dropping of device objects.

5.3.4. Draw Eraser

This scenario is of a user who is trying the Microsoft Paintbrush package for the first time. The user is already familiar with MacDraw. The two packages have a similar look and feel. A palette is located to the left of the draw-space, and a menu-bar runs horizontally above it. The basic manipulations are broadly similar. The user, however will not be familiar with some of the icons displayed on the palette. The user wants to

create a composite object, involving several line and shape objects. The familiar look and feel enables the user to recruit general rules of operation from the MacDraw rule-base. For example, a range of palette icons are recognisable as equivalent to the MacDraw icons. Therefore, search and the specification of operations are heavily influenced by knowledge of MacDraw.

However, there are some differences between the two packages, which affect some tasks. An example arises when the user wants to delete component items from the object, including part of a line. The user has seen the 'pencil rubber' icon on the palette (a feature with no equivalent on MacDraw). Therefore, the user has to translate known principles or heuristics from the domain to specify operations. The cue expresses an 'erase' metaphor, which binds to the rubbing out of pencil drawings. The user is prompted to try similar actions with the mouse, thus learning a new utility.

5.3.5. Editing Drawn Objects on a Document

In this example, the user wants to move a labelled box. After forming the goal the user distinguishes the two entities (i.e. the text item and the shape item) which form the 'labelled box'. In order to optimally perform the action, the user must match moving a composite object with the lasso action. This does not involve a selecting action in a conventional sense, but the placing of the cursor in a neutral area of the screen and dragging to enter lasso mode. The cue is a 'hand' which appears if the mouse button is pressed on an object-free region of the draw-space. On dragging, the hand becomes base node for a feint line which expands and contracts according to where it is dragged. Subject A in the Chapter 4 study found this feature by noticing feedback from an accidental action. The feedback showed several drawn objects with the familiar 'handle' dots (showing that they were selected) around their edges. This is an example of an interpretation rule being employed. Subject A was able to use knowledge of the package to comprehend the feedback, having seen selection 'handles' for single objects. The feedback thus became a cue for the next action.

5.3.6. Selecting from Menus

Menus are an example of a two-phase search. The user has first to select a search location. The second phase is the scanning of that location for an appropriate feature. In this example the user has knowledge of MacWrite. The user's goal is to create an identical object to one that is already drawn. As it is a complex composite object the user wishes to make a copy automatically. The user sees an Edit menu. The user is able to link this familiar cue to the type of utility that is needed (Copy and Paste facilities are located on the MacWrite Edit menu). The user pulls the menu down and

scans. The Copy and Paste options are seen, and are assumed by the user to be similar to the MacWrite equivalent. However, the user also sees a Duplicate feature. The user recognises the cue as relevant to the current goal by matching the term to the task-action. Therefore, a different knowledge-space is used for feature selection than for the original search. The user uses knowledge of operational rules to operate the feature.

This example shows that the guiding of search and prediction of feature operation can be separate for some actions. The visual metaphor may indicate where types of feature are likely to be. Recognition of the feature itself may result from a different knowledge space. In turn, recognition of a function does not imply knowledge of its operation. The user recruits knowledge of how to perform a menu operation on specific objects.

5.3.7. Summary of Scenarios

The scenarios demonstrate variations both in knowledge-spaces that are used, and ways in which they are utilised. The Wastebasket scenario demonstrates a feature cue prompting contrasting knowledge recruitment by a complete novice and a novice with some package knowledge. The two users were able to infer the necessary operations by different means. The user in the 'Eraser' scenario needed to comprehend the 'eraser' metaphor to learn the mouse and pointer equivalent of 'rubbing out'. These examples show that different knowledge-spaces may be accessed for search than for specifying operations.

The Lasso scenario shows a contrast with those described above, in that there is no search as such, but simply recognition of feedback, to facilitate trial and error learning. Both recognition of a feature and an operation are made clear by evaluation of the system state. This contrasts with the 'duplicate' scenario, in which a location was selected using device knowledge, and a feature recognised using scanning and matching to the task-space. The user subsequently employs a known rule to operate the feature.

5.3.8. Implications for Evaluation

This section describes the knowledge-spaces that influence user behaviour. The system metaphor and individual feature representations may prompt the use of different knowledge-spaces dependent on the users' experience of both task and device. This has implications for the evaluation of dialogue design. If a package is to be usable for a wide range of users, the design must support both knowledge-based

and rule-based processing. This applies both to high-level organisation and the design of individual functions.

Knowledge-spaces may be used in locating (selecting where to search for a feature), recognising an object (feature selection), and working out the necessary action (specifying the operation). Visual aspects of the interface may have a role in supporting one or more of these mental activities. Therefore, evaluation should pinpoint a dialogue design problem as a failure to support one of these activities. It is useful to analyse critical incidents both by analysing the users knowledge recruitment in response to environmental cues, and by the role that the interface has in supporting a mental activity. This requires unpacking the 'specifying an action' stage in Norman's (1986) theory of action. This stage may usefully be seen as a process of locating features, identifying features, and specifying operations. These are considered separately because a different element of the design may be responsible for a critical incident. The study of MacDraw II in Section 4.6. suggested difficulties in accurately diagnosing a dialogue design problem.

5.4. Error Diagnosis in the Context of User Activity

5.4.1. Overview

In the original model in Chapter 3, error types are attached to points in the cycle where user behaviour is caused to change from normal task action. In this section error types are traced to their source. This uses the distinction between the observed effect of an erroneous action (an observed or reported error) and the root cause of the problem described by Hollnagel (1993). In Hollnagel's analysis, observed user errors are described as phenotypes. All phenotypes are connected to, but distinct from, genotypes. Genotypes are defined as 'the functional characteristics of the human cognitive system that are assumed to be a contributing cause of the erroneous actions'.

The phenotype/genotype distinction helps to clarify the process of evaluation using error analysis. Error-based evaluation has three basic phases, namely error discovery, error diagnosis, and solution specification. In an evaluation process (e.g. protocol analysis) the phenotype is the 'discovered' error observed and/or reported in an evaluation session. Diagnosis is the process of determining the genotype, and solution specification is the result of analysis of one or more genotypes. More specifically, solution specification is conducted from a work context, determining the system's (or a system feature's) contribution to a genotype, and how (if at all) a design modification may remove the problem. Also, it is possible that identical phenotypes observed from

different users may have differing root causes. One user may fail to operate a feature correctly because the feature metaphor is inadequate. Another user may have the same overt problem as a result of generalisations made from previous interaction with the package.

The amount of analysis required to connect phenotype and genotype may vary. In some cases, the genotype is clearly observable (e.g. the user has a problem continuously monitoring responses to dragging actions, because the cursor is not moving in the expected direction). In other cases, the genotype is a more distant cause, rooted in a previous action with a feature or the user's understanding of a key system concept or term. Examples of 'distanced' causes of user errors are described by Carroll et al (1993).

The following sections link genotypes diagnosed in the Chapter 4 study to stages of the Models of Action described in Section 5.2. Errors linked to types of processing are described, linking phenotype to user genotype. Some are illustrated with sample protocol extracts from the Chapter 4 study to demonstrate the link between phenotype and genotype.

5.4.2. Knowledge-Based Processing Errors

5.4.2.1. Guess Location Phase

Two types of error are associated with the mental act of guessing a feature location. These are the inability to select a search location, and the incorrect selection of a location. Knowledge-based search relies on the visual image to give the user an indication of how the screen is organised. In the absence of such assistance the user may search locations (randomly or in an arbitrary sequence) or give up. The genotype may be described as metaphor or structural layout failure, in that the user is given no navigational assistance.

The user may reason about the search location from the visual metaphor, or read menu names. The phenotype is the user reasoning in error that the feature is in a certain place. The genotype is either confusion over the meaning of a section name or representation, or the presence of a feature in an inappropriate location. The following extract is of Subject E searching for a facility to change text point sizes, to fit a text item into a drawn shape. The extract is divided into physical (actions) and verbal (utterances).

Verbal - *I'll try to make this (text) smaller*

Physical - Goes to Layout Menu, scans

Verbal - *That's no use*

Physical - Goes to Edit menu, scans. Goes to Layout menu, scans again, to Arrange menu, scans.

Verbal - *That's no help, I'll have to make the circles bigger. It's going to be a tight fit.*

Subject E had to alter the diagram because he could not find the point changing facilities. He looked in (apparently) appropriate menus (Edit, Layout, Arrange), using the header names to help (point options are at the lower end of the 'Font' menu).

5.4.2.2. Select Feature

Two errors associated with recognition are failure to recognise a feature and the incorrect selection of a feature. In the former case, the phenotype is the user passing over a needed or useful feature. Broadly, the genotype is the cue failing to alert the user to the feature's utility. This may be because the user cannot connect the feature representation to the current goal.

There may be a problem of prominence or visibility, with the user scanning and not noticing an item is there at all. The feature cue may be too small to see, or not cued at all (e.g. the lasso).

The phenotype associated with incorrect feature selection is the user reporting failure of an action, and incorrectly identifying a feature. The genotype is the cue incorrectly suggesting a link between the feature and the current task-goal.

Erroneous choice of features may be linked to interface support for the location of features. For example, the names of utilities in a menu may not be distinct outside the context of the group header. This is illustrated by an example transcript of subject C searching for facilities to reduce the size of a line object.

Verbal - *I'm going to have to be a bit careful how I use the rest of the space on this. I'm slightly running out. I've only considered putting it down as I want it to look. it would be a pain putting lines in and then deleting them. I presume you can't squash lines.....I don't know*

Physical - Goes to Arrange menu , Goes to Layout menu

Verbal - *No, I can't see anything. wait a minute .....'reduce to fit'*
....I'll try that

Subject C incorrectly matched the goal of reducing a line to fit a space with the 'reduce to fit' cue. This demonstrates the way in which a bogus match can result from ambiguous cueing. It should be noted that he was clearly unaware of the general function of the menu, which offers options on the viewing and function setting of the draw space.

5.4.2.3. Guess Operation

Two error types are associated with the 'guess operation' phase. These are insufficient knowledge of needed operations, and finding that functionality is inappropriate. The user may not be able to work out how to operate a feature. For example, s/he may not realise that certain steps are required, because they do not map to the task-space. For example, Subject A found that he could not plot a point with precision. The following passage is taken from Subject A's session in the Chapter 4 study.

Physical - User selects palette default. User moves cursor to text within a circle. Moves the first line a small distance. Repeats for two more lines.

Verbal - *I should be able to move and position it. Its jumping about a bit. I'm surprised that such a powerful package does not let you do it accurately*

Physical - Selects palette text option. Moves the cursor to a (drawn) square. Moves cursor again and clicks to confirm location.

Verbal - *I press the mouse there to indicate that I want it there. It flashes below the right place .....*

Physical - Moves cursor slightly higher and confirms location

Verbal - ..... *and then at the right, top, or left of the mouse. I find that very confusing. It means I have to learn it before I can use it effectively*

Unknown to him is that the default state of the package is an invisible plotting restriction (or 'grid'). This principle is missed because it is unexpected and counter to the 'drawing' metaphor. Subject A remained unaware of the 'Turn Grid Off' feature. He was unable to recognise the feature when shown the menu cue in a post-session test. He admitted that it had not occurred to him that such an option may have existed.

This error is an example of a failure to support the natural actions. The user's estimation of the optimal sequence of operations is drawn from knowledge of pen and paper drawing. The user has no concept of a grid, and is therefore unable to specify operations satisfactorily or interpret the system response accurately.

The user may find that a feature is simply incapable of operating as desired. The phenotype is the user expressing a problem with the way that a feature has behaved, despite having identified the feature correctly. The genotype is the feature failing to support the precise goal. This is usually because the feature is too restricted. For example, the cut and paste facility may be operated to move an integrated text and shape item. However, the system may not allow the simultaneous moving of both text and shapes. The feature has been correctly recognised, but its scope of operation is too limited. The following transcript is of Subject D trying the Freehand Draw option from the MacDraw palette. He trying to draw a line connecting two objects whilst avoiding an obstructing object between them.

Physical - selects Freehand palette option, draws line - deletes - repeats action - deletes

Verbal - *It's a bit of a bummer this thing, because you have to press and at the same time have very good poise to draw lines. I wouldn't recommend it. but I don't have much choice because there is no other choice. Of course if I don't have to follow this thing I can use straight lines, although its not as good.*

Physical - Selects Freehand palette option, draws line - deletes - repeats action - deletes

Verbal - *So it looks like I have to use straight lines, because I don't think thing is actually a connection. it looks like it is just something to scribble with.*

Subject D identified the feature correctly as a freehand drawing option. However, the feature is not capable of matching the performance of a pencil in terms of accuracy and

neatness. Subject D had assumed that the feature would support skilled drawing.

5.4.3. Rule-based Processing Errors

5.4.3.1. Goto Familiar Location

The user will have rules for governing search, linking areas of the screen with types of feature. The user may go to a location to look for a feature, but not find the feature. The genotype is the failure of the user to predict the feature location using experience of feature search. For example, the 'show clipboard' feature on MacDraw is located in the File menu, although the Cut, Copy, and Paste options are in the Edit menu. The following extract describes Subject D searching for a 'clipboard' facility to check the contents of the buffer.

> Verbal - *I don't know if all of it copied. I'll look at the clipboard.*
>
> Physical - Goes to Edit Menu, scans.
>
> Verbal - *It's not here. Why isn't it here. It should be. Every system I use lets you look to see what you copy..... I'll just do it and hope it's o.k.*

Subject D admitted that he had searched for the clipboard, and had given up after looking in the Edit menu ('show clipboard' is in the File menu). He had assumed that the feature would be clustered with these relevant options. It retrospective interviews he cited experience of the Edit menu on MacWrite, where the clipboard feature is clustered with Cut, Copy and Paste options.

5.4.3.2. Retrieve/Modify Operation

This refers to the mistaken use of abstraction-based reasoning. The phenotype typically involves the user expressing expectations from previous system actions, often about the behaviour of features that appear similar. These expectations are confounded when action is attempted. The genotype is the user making abstractions and generalisations on the basis of visual or conceptual associations between features, and those differing from the system model. In other words, the user misapplies general rules, having been cued to do so by the system. The following transcript sample describes a particularly serious error of this nature from Subject H. He has selected other drawing facilities from the palette, and has experience of adding arrows to lines.

Verbal - *I will now attempt to do one of these curvy lines*

Physical - Selects arc palette option - draws arc - goes to lines menu

Verbal - *I'll put the arrow head on it*

Physical - Selects arrow option (no response)

Verbal - *I don't know why it didn't do it.*

Physical - Selects arrow again

Verbal - *do I have to set it onto a line?*

Physical - Reselects drawn arc

Verbal - *I think I've selected which lines I want to put the arrows on, that's what the dots are. It should put arrows on there..... .....oh well, I don't understand this one. I put on the arrows thing- so it should give me an arrow. I identify the the line I want to put it on, and I go to the arrows..... it should draw it, but it doesn't. What do I do if I can't do it?..... I'll try something else.*

Physical - Selects diagonal line palette option, draws line, goes to lines menu, selects arrow (arrow appears)

Verbal - *ah right! so you identify the arrows you want and then draw the line.*

Physical - Selects arc, goes to lines menu, selects arrow option (no response)

Verbal - *I drew it once before- it's not working now*

The system has failed to support abstraction-based generalisation by the user. The user has applied a known operational rule to a new feature, assuming that it is a feature of the same type. Therefore the user assumes that the same operations can be applied.

Another error affecting rule-based specification is hidden effects of a previous action. The phenotype will typically be the user expressing strong expectations about the

result of an action, and some bewilderment at the result. The genotypes are the effects feedback perception mismatches from a previous action. This genotype is also linked to mode errors, described by Norman (1988) and Monk (1986). In the example below, Subject A has set a default without realising. The default automatically adds arrows to any drawn line. He had been drawing arrowed lines by selecting a palette option followed by an arrow option from the lines menu, and lastly dragging the line. By selecting the line option before drawing the line, he has selected a default arrow which is activated for all subsequent horizontal and vertical lines. He is trying to create a datastore (i.e. three sides of a rectangle) using the horizontal lines option. It is around ten minutes since he last drew a line.

Physical - selects horizontal line menu option - draws line

Verbal - *It's, in fact, still drawing those arrows. I'll have to reselect. It's not exactly playing fair*

Physical - Goes to lines menu - selects 'no arrow' option - selects horizontal line menu option - draws line (arrow appears)

Verbal - *Its gone back to the arrow. Now why has it done that? That is totally baffling to me. Why on earth have I got an arrow on that.*

The user has no knowledge that a default is set. Therefore the user has made incorrect assumptions about the state of the machine in specifying the action. The default has been set, but the user was unaware that this has been done. This will be further discussed in the section below on feedback.

5.4.4. Skill-Based Errors

The problems of search, recognition and specifying operations do not apply to skill-based action. Skill-based action uses compiled procedures rather than recruitment and use of knowledge. Therefore, associated errors are likely to be those associated with the tight cycle of skilled performance and continuous monitoring of progress. In highly interactive systems the user may become able to perform manipulations extremely swiftly. This may lead to problems if the user becomes sufficiently fast that the dialogue becomes restrictive. Another danger is that indistinct cues may trigger the wrong skilled procedure. Also, with frequent use, monitoring of system responses may become less precise, or even stop altogether. Therefore the user may perform sequences too quickly for the system to validate the input. Of the eight incidents involving subjects failing to confirm a palette selection prior to commencing drawing,

seven were made by two subjects. The errors all occurred towards the end of the session when subjects began making the error after using the palette on numerous occasions for repeated operations.

5.4.5. Summary of Errors

The examples demonstrate a diversity in error types between processing levels. However, a number of design problems are similar or common across processing types. For example, problems with interpreting the spatial metaphor may prevent the user from effective searching (knowledge-based processing error) or lead to a more experienced user selecting the wrong search location and missing a feature (rule-based processing error). In both cases the mental act of locating features was not adequately supported by the system.

System genotypes can be traced by eliciting the user's interpretation of the interface, and the knowledge-space that the user employs for a particular mental act. As shown in Sections 5.3. and 5.4. users may recruit from different knowledge-spaces for different mental acts within a single action. Each mental act is system led, in that the user is relying on signals from the interface. Therefore system support for each user mental act is a potential source of an error.

The error types described above deal specifically with mental acts involved in the specification of action. The following section adds the remaining stages of action including recognising and evaluating changes. Recognising change is simply noticing that a change has occurred. Evaluating change is the act of interpreting and comprehending the change. There are also problems associated with the initial formation of goals. Also, errors may be associated with the actual physical input actions. These are discussed in the following section.

5.5. Errors in the Recognise/Evaluate Change Phase

5.5.1. Recognition of Changes

The errors discussed here are problems in noticing that changes have occurred. The phenotype is the user expressing that an action appears not to have had an effect. The user may have inadvertently activated a feature, and the phenotype is a subsequent reaction to unexpected system behaviour. If a change is not indicated with feedback, the user may specify action with an incorrect model of the system-state. Equally, a delay may cause confusion, as the user will rapidly proceed to the next action, failing

to notice. Similarly, feedback may be too inconspicuous, brief or indistinguishable to alert the user.

5.5.2. Evaluation of Changes

This refers to the accuracy of the system image after an action has been performed. The phenotype will typically be the user believing that there is a problem, even though the correct procedure has been followed to achieve a goal. The genotype is the system failing to give an accurate visual indication of a state change. For example, a delete action may seem not to have worked because an image of the deleted item (or a remnant of it) remains on the screen.

Another problem is failure to interpret feedback in task terms. A change may occur as a result of an action which fails to indicate the system state. The phenotype is the user recognising but misinterpreting a change. The genotype is a failure to match system output to user goals. For example, the object selection action in MacDraw produces 'handles' on lines or shapes. This represents a state where moving, reshaping and resizing are possible through direct manipulation of the object. However, the representation is arbitrary, and may not be understood by the user. The user may misinterpret feedback, and be prompted to perform a wrong or unnecessary action. For example, placing text objects on existing shapes (in MacDraw I) may result in the potentially misleading appearence of a partially deleted shape. The user may be prompted by this to delete one or both objects unnecessarily.

5.5.3. Formulate Goal

The declaration of a user goal implies the need for system support. The user declaring a goal which is beyond the scope of the system is therefore evidence of a system error. The phenotype is the user searching in vain for suitable features.

5.5.4. Execute Action

The 'execute' action phase places physical demands on the user. The user may find that an action is simply too difficult to perform. The phenotype is the user experiencing difficulty performing with a feature to a sufficient level of satisfaction. For example, the user may need to perform dragging or pointing actions with extreme accuracy and find the system does not assist the user's physical capacities sufficiently. The genotype is a failure of the system to adequately assist the user's (limited) manipulation skills with support functionality.

Another problem with execution of action is unnatural response to user input. The phenotype is the user finding that a feature responds in an awkward way to user input. For example, some scroll-bars allow the user to click and drag, but scroll the text at a considerably faster rate than the pointer moves. The user is unable to track the scrolling action. The genotype is the failure of the system to support the user's motor skills. It is possible, with practice, to master the operation. However, learning of such skills should not be necessary. Manipulation and the device response on-screen should co-ordinate in a natural way.

5.5.5. Discussion

The examples show the need to analysis the deep causes of critical incidents. The phenotype is the user indicating a problem. The feature and task involved may be indicated both by observation of action and by verbalisation. Also, the user may give an indication of the mental act that was the source of the problem. In other words, system failure to support a particular mental act may be seen as a failure to play the relevant role in the dialogue. The user relies on the interface to support mental acts, so system support for a particular mental act may be thought of as a particular dialogue role. This is discussed in more detail in the next section. Along with features and roles, the sources of users' reasoning is indicated. This is further contributory evidence indicating why a role failure has occurred, and the design attribute that may require attention.

5.6. Dialogue Roles

5.6.1. Introduction

Diagnosis by interpretation of user errors may pinpoint a feature involved in an erroneous action, without pinpointing the actual problem that it actually caused, or indicating how designers may rectify the the problem. Accurate diagnosis requires analysis of the role that a particular system feature plays in the user problem, a point made by Booth and Gray (1990). This implies the need to understand the role that a design feature should have played in the dialogue (i.e. the mental or physical act that required support). Serious user errors may often result from remote or high-level causes rather than the feature 'implicated' by the phenotype (see Carroll et al 1993).

The approach described in this section is to define an abstract set of roles which are common across a range of task-actions. These correspond to user mental acts in the cycle of action. Figure 5.5. shows dialogue roles linked to knowledge-based, rule-based and skill-based mental acts. The sequence of Dialogue Roles is now described

with reference to types of role failure.

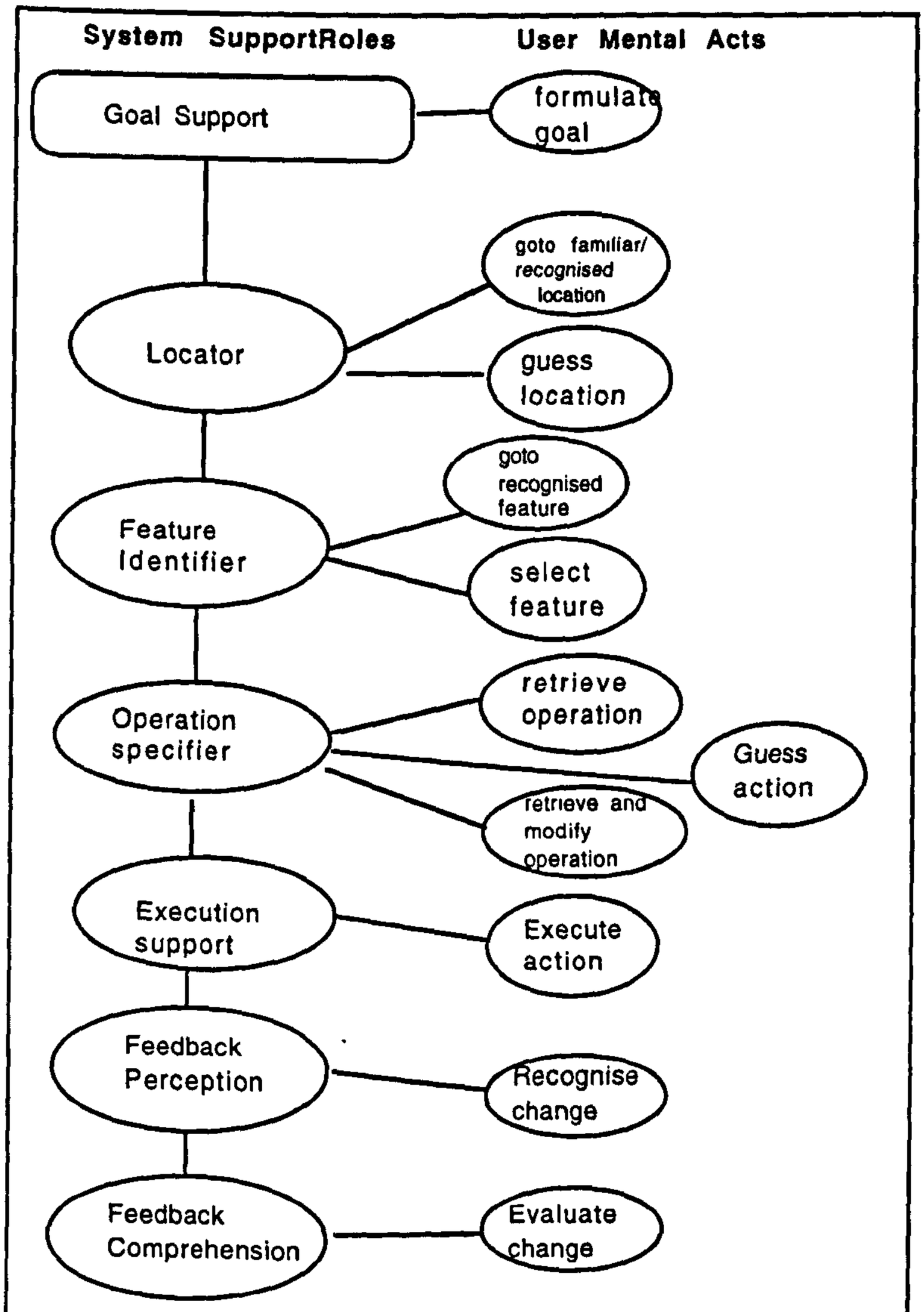


Figure 5.5: Dialogue roles linked to user mental acts

5.6.2. Dialogue Roles and User Mental Acts

5.6.2.1. Goal Support

The user reasons about the domain, what is desired and what is likely to be achievable with the device. The dialogue role here is simply supporting the goal by providing appropriate support for the task. The associated role failure is the absence of functionality with which to achieve the goal.

5.6.2.2. Locator

Some actions require two phases of search, including the initial selection of a search location (e.g. search within menus). The dialogue role implied here is the Locator. The locator role refers to the information which informs a user where to search for a feature. For knowledge-based processing this means support for the 'guess location' mental act. This may be the adequate visibility of an area, a comprehensible system metaphor and/or semantically relevant labelling of areas and menus. For rule-based processing (goto familiar/recognised area) the requirement is broadly similar, but with particular emphasis on the placing of supporting and and advanced features. The rule-based user is likely to have a strong model of the spatial layout gained from previous interaction, and be less inclined to perform a general search. Therefore, features which are inappropriately placed may not be found.

Role failures associated with locators may be a failure of the visual and logical layout to allow the user to identify locations. The inadequate naming of menu bars to assist scanning is another example. Also, the inappropriate location of a feature may cause the user to search in the wrong place. It is possible that a user may select the wrong feature because a cue is understood in the wrong context. Whilst this may also be a feature identification problem (see below) it may be, for example, a failure to label feature grouping adequately. Also, the user may select an 'obvious' location, but the feature may be elsewhere. Figure 5.6. shows examples of Locator problems found on the MacDraw menu bar.

5.6.2.3. Feature Identifier

The user scans for a feature which has characteristics matching the user's goal. This may be seen as matching a device representation with some element of the task model. This can be illustrated using a description of novice DM users trying to identify a menu option, from Young et al (1990).

> '.....the novice has to proceed by seeking inspiration, as it were, from the information present on the display. The user will have to ponder the candidate items in turn, and ask of each "given what I know about the task,

can I see any plausible connection with this word, strong enough to make me want to consider selecting it?'

To be a little more specific, the user is searching for a mapping between task concept and device feature. The success of the cue is gauged by the strength and distinctiveness of this connection. This may be a metaphor suggesting a task concept (knowledge-based), or a familiar name, such as a feature which is familiar from previously-used packages (rule-based processing). A similar process applies to the scanning of other interface objects, such as metaphorical representations of domain objects. In these cases, the user simply scans a candidate location rather than a range of possible options. The role implied is that of the 'feature identifier'. The role is to provide a semantically relevant token, which is clear and distinct to the user. The representation must cause the user to connect the feature with a current goal.

The role failures associated with this role are lack of cue meaning (i.e. lack of semantic attachment between feature representation and task model concept) lack of cue salience, and misleading cues. In the former case a cue may refer to a system rather than a task concept, or simply not be a commonly recognised expression. Also, there may be ambiguity, where a user makes a false connection and selects the wrong feature. Some feature Identifier problems are shown in Figure 5.6.

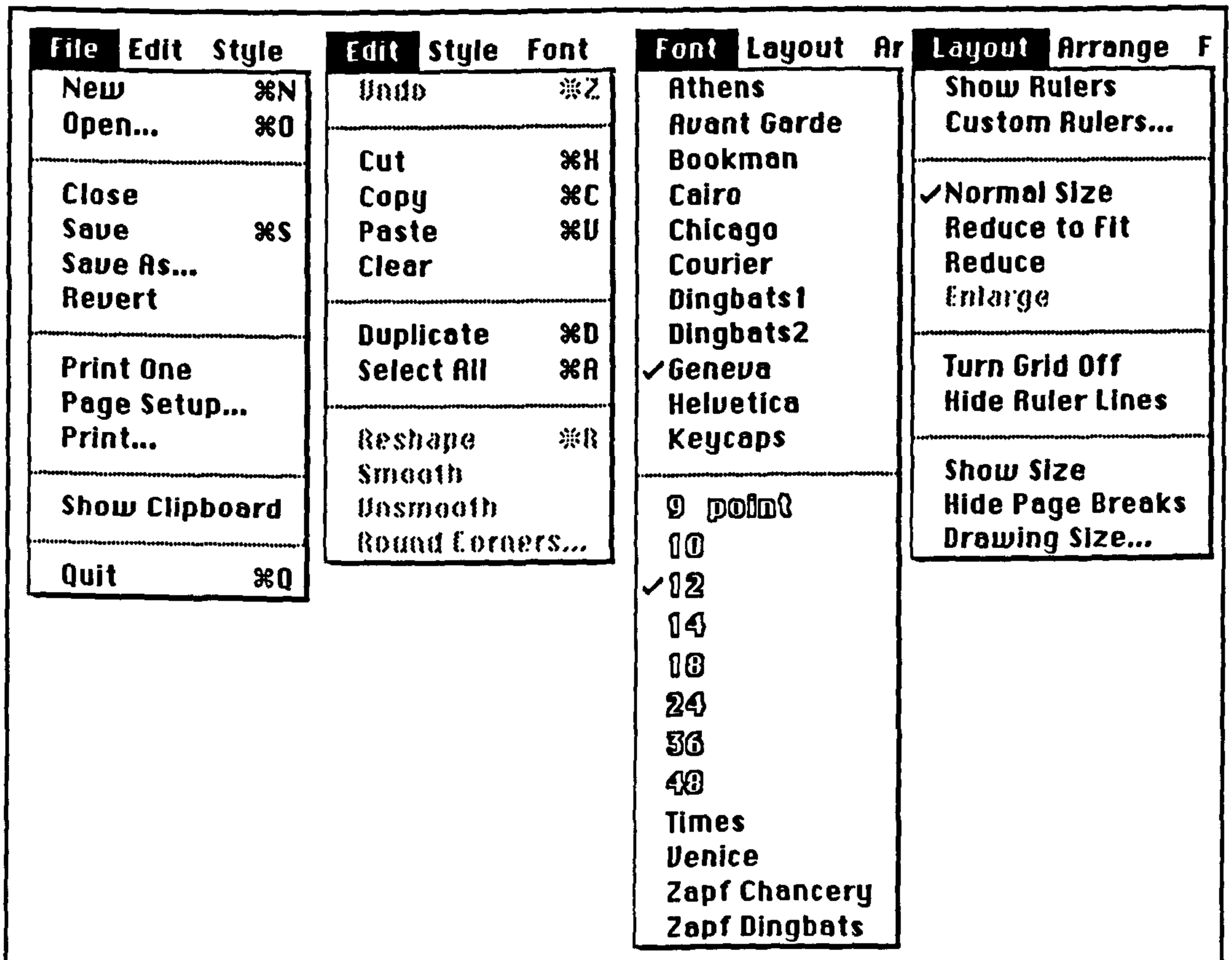


Figure 5.6: Locator and Feature Identfier Problems in the MacDraw menus. The Show Clipboard feature is in the File menu, separated from the Cut, Copy and Paste options (Locator). The Font menu also has point changing options, not referred to by the menu header (Locator). The Layout menu shows the 'Reduce to Fit' option which subjects tried to use to alter text and object size (Feature Identifier). Also, the feature 'Turn Grid Off'was scanned but not recognised or understood by subjects (Feature Identifier).

5.6.2.4. Operation Specifier

This phase describes the user's reasoning about the device operation. This type of activity is particularly dependent on the nature of the user's prior knowledge. A user may recognise the selected feature as being of the same type as one previously used ('retrieve and modify operation') and apply a known rule. However, the process may involve knowledge-based processing. The user may use knowledge of the metaphor to imply necessary operations ('guess action'), or simply guess.

The relevant dialogue role is the 'operation specifier'. This refers to system support for the selection of appropriate operations. The nature of the operation specifier role varies with the type of processing that the user is employing. Where the mental act is 'retrieve operation' and a feature and operation are re-used (skill-based processing), the role implies consistent operation of the feature. So if a feature operation changes when the system is in a different mode, that difference should be clear to the user. Where the mental act is 'retrieve and modify operation', and generalised operational knowledge (rule-based processing) is used, the role implies consistency of operation across features. This includes grouping features in a way that cues similar operations, and communicating any operational contrasts. Where the mental act is 'guess operation' and the user is reasoning using their model of the task (knowledge-based processing), the sequence of device operations should be predictable.

Role failures, for re-used features and procedures, tend to be mode changes or other system state changes effecting the feature that the user has not been made aware of. In the case of analogy with other features being used, role failures are the result of inconsistencies in the way that features operate (e.g. method of activation, legality of operations). In the case of reasoning using the task model, unexpected extra or omitted steps are likely to cause problems.

5.6.2.5. Execution Support

Execution refers to the physical act of operating the device. The execution role involves support for the users physical skills. This is 'natural' system responses which minimise the need for the user to learn device-specific skills.

An example of an execution support role failure is found with some scroll-bar facilities. The system begins by moving at a comparable speed to the user's mouse action. However, the scroll may suddenly accelerate, making it harder for the user to control. This could be considered an unnatural system response.

Another potential problem is that the needed action may be too difficult to perform. For example, the accuracy of a required drag/drop action may be beyond the skill of a user. The user may not be able to see or reliably move the mouse with the needed accuracy. The designer should either design manipulations to be within the normal ability range of individuals (e.g. constrained drawing options for dragging straight lines), or add support options (e.g. a zoom for unusually precise plotting actions). The absence of sufficient support may be considered a design error. Also, the adequacy of a feature for the user's specific needs may be a problem. For example, a rotate feature for drawn objects may be needed for a number of rotation angles. If it is found to be restricted to 90 or 180 degrees, users may find this inappropriate to their specific needs.

5.6.2.6. Feedback Perception

The user, if the action is novel, may feel the need to study feedback closely. However, this activity is often a brief scan, or particularly with familiar actions, a continuous tight sequence of automatically performed actions (characteristic of direct engagement).

In a number of DM interactions, actually providing a signal which draws the user's attention to a situation may be difficult. This is particularly so because even novice users actually perform device actions extremely swiftly. Where actions are repeated, the temporal gap between one atomic action and the next may be negligible. In this example, the Feedback Perception role is to provide confirmation which enables swift progress to the next atomic action.

Feedback perception role failures refer to feedback which is absent (not provided), delayed, obscure or too brief. Related to this is the problem of secondary or remote effects of an action. The user may confirm an action unaware that a secondary effect has occurred. The manifestations of this change may only be visible when a subsequent action is performed, lessening the likelihood of the user being able to comprehend the system state.

5.6.2.7. Understanding Feedback

The user matches the system state to expectations and goals. An important consideration for evaluation is the user's perception of progress towards completion. This is particularly the case where a user action has a duration, as is the case with dragging actions. An error described by Sutcliffe and Springett (1992) emphasises this point:

'When users drag a shape over another one, their expectation is that the first (moved) shape will still be visible inside or underneath the second. The feedback during the dragging suggests that the shape is transparent (the text and ruler lines are visible underneath). However, on releasing the mouse-button the shape becomes 'solid', obscuring the object underneath. Even when action is specified and is being performed, feedback from the interface should provide the user with information which determines the next mouse movement, or a modification of intended actions'.

The feedback should allow the user to reference the state change in terms of his/her notion of the goal-state. This is particularly pertinent were user action has remote consequences, such as the printing of a document or updating files which are not open on-screen. In using device independent knowledge in the formulation of action, users also create expectations about the appearance of the post-operation state.

Feedback comprehension role failures are failures to reflect the true effects of the action, and/or failure to reference user goals. The interface presents an image of the end-state of an action which is open to misinterpretation.

5.6.3. Using Roles in Critical Incident Analysis

5.6.3.1. Introduction

The previous sections in this chapter refer to the need to establish genotypes from the analysis of phenotypes or critical incidents. This section describes the way in which pinpointing system failure to fulfil a role (i.e. failure to support a particular mental act) contributes to this process. The failure of the system to fulfil a role will henceforth be referred to as a role failure.

When a phenotype is observed, the evaluator has the physical evidence of a user struggling with a task. This may be augmented by a verbal description from protocol analysis, giving an indication of the user's expectations, reasoning, and interpretation of the interface. At this point the evaluator has some notion of the cause of the problem. A particular task and feature will be implicated in the critical incident. This may be a user unable to proceed, having difficulty in performance, or expressing dissatisfaction with the result of an action. In all cases (except where the user is at a complete loss) the user will be able to express an intention. The user should be able to express their interpretation of general and feature level metaphors, and their reasoning behind search and operation specification. This information can help pinpoint the role failure in the current action.

The raw evidence of a phenotype will show the evaluator that a user finds a particular task difficult or confusing. However, this leaves the question of appropriate design changes open. Identifying the problematic aspect of a feature (or design concept) is the crucial step. Earlier in this chapter, the varying contributions of visual metaphor, feature cues and general operational principles were described. Each single contribution represents a potential role failure. The following sections describe ways in which phenotypes from observation and protocol analysis may be examined to pinpoint role failures. Phenotypes are are distinguished by whether they occur before during or after a user action. Figure 5.7. shows the link between phenotypes, supporting evidence augmenting phenotypes, and types of role failure. The left-hand shaded area shows the supplementary protocol evidence that can assist deeper analysis. These are linked to the right-hand column which show the types of role failure implied by the evidence.

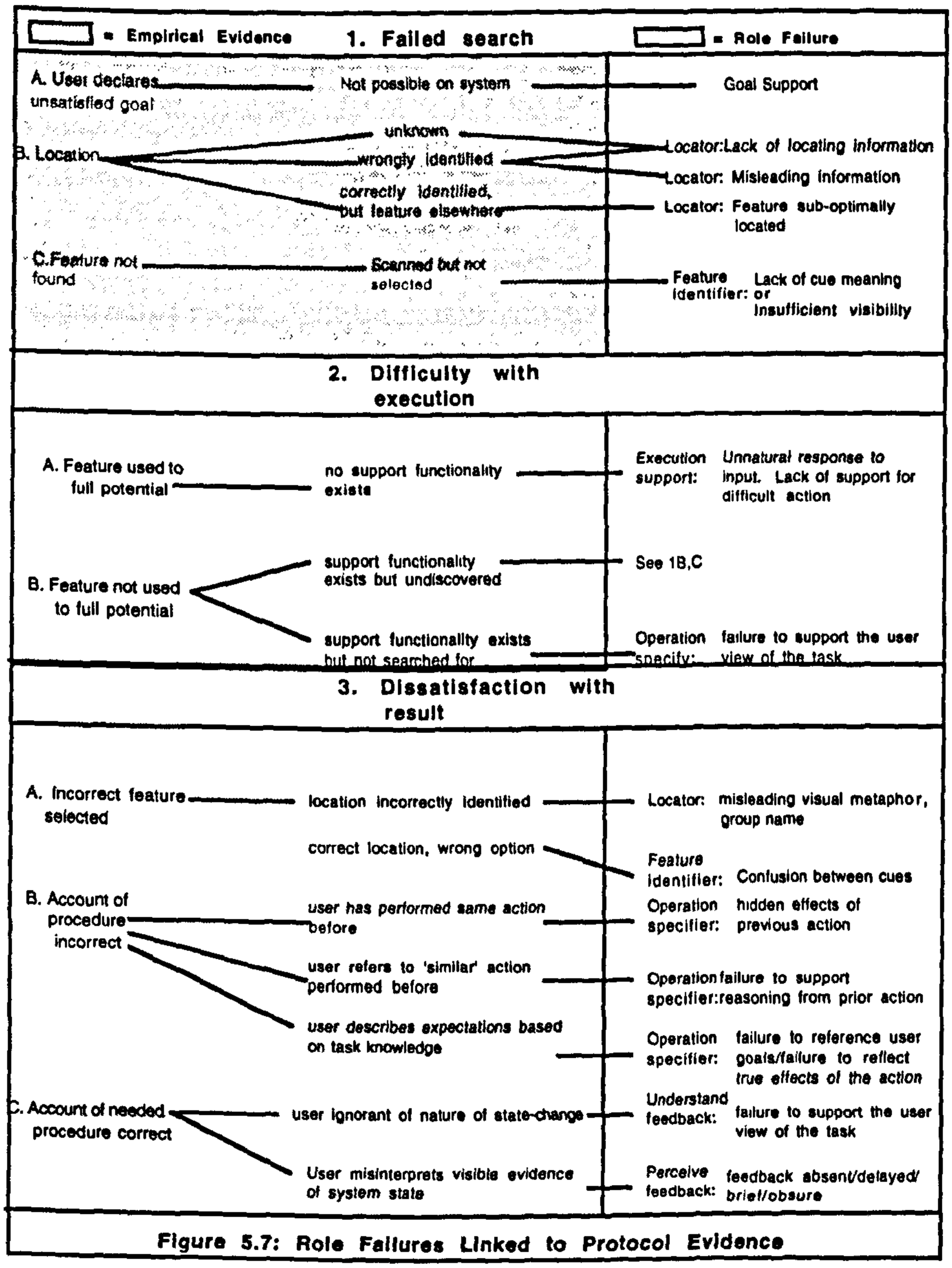


Figure 5.7: Role Failures Linked to Protocol Evidence

5.6.3.2. Failed Search

The user may fail to find a feature although reporting the goal, the needed utility and the extent of the reasoning behind the search (if any search has been attempted). The evaluator can infer whether there is a 'goal support' role failure, by simply assessing if the needed functionality is provided by the system. If it is not, there is evidence for adding such functionality.

The user may search for a feature, scan it, but not select it. This suggests that the cue is either not sufficiently visible, or does not have characteristics that effect recognition (feature identifier failure). This provides evidence for the need to alter the cue. Also, the user may describe the type of feature that they are looking for. This may provide an indication of what the new feature representation should refer to.

The user may fail to direct search for a feature, scanning features randomly, or simply declaring frustration at the absence of a feature. This suggests that the visual layout of the system is not understood, as the user is unable to link a feature type to a location. Also, the user may declare a strong belief that a feature is in a particular location, when it is, in fact, elsewhere. This may provide an indication of the way in which the user interprets the spatial metaphor. In turn it provides evidence in favour of altering the grouping of features or the location of a particular feature.

5.6.3.3. Difficulty in Performance

The user may encounter problems in performing an action. The user may either describe what was expected (e.g. 'the cursor should move the other way') or difficulty whilst the action is performed. The evaluator should judge whether any of the existing facilities on the system could help the user. If they exist, but the user is unaware of them, that suggests the problem lies in the specification stage. The user may have no idea that, for example, a supporting feature exists on the system. This suggests a failure of the 'operation specifier'. The user may have scanned the interface for the feature and failed to find it. This could be a locator role failure (if the user has not gone to an appropriate location) or a feature identifier role failure (if the user has seen but not recognised the feature).

5.6.3.4. Dissatisfaction with the Result of an Action

Ideally, the user will provide both an interpretation of the system feedback and an expression of the goal and prior expectations. The user may indicate a misunderstanding of the system state. This may be a mistaken belief that the goal is not satisfied, or a mistaken belief that the action has not contributed to the satisfaction

of the goal. The user's description of what happens may show that feedback has either not been noticed or has not been understood. In the latter case the problem may simply be that the feedback fails to reflect the task state adequately. However, it may also reveal a specify operation role failure. The user may have mistakenly believed that the actions already performed were sufficient to satisfy the goal, revealing ignorance that further user actions are necessary.

The user's account of expectations may also reveal the incorrect choice of a feature (failure of the feature identifier role). Also, the user may express expectations based on previous interaction which have not been supported. This would be a 'Specify Operation' role failure, defined as a failure to support assumptions from previous action. Also, the user may express ignorance of the result of a previous action. This would suggest the possibility of a problem with the design of feedback in the previous action. A change in the system state may be unknown to the user, whose protestations may suggest the belief that the feature seems inconsistent.

5.6.4. Summary of Roles

The diversity of evaluation causalities is a potential problem for novice evaluators. Analysis using roles can prompt evaluators to scan a range of possible genotypes, and check the validity of their 'intuitive' problem diagnosis. In turn, it can be used to guide evaluators to the cause of an error, including causes that lead back to prior interaction or high-level problems. Analysing roles allows evaluators to pinpoint not only features, but properties of features and their contribution to dialogue. Also, the analysis allows the contribution of both feature level concepts and higher level metaphor concepts to be assessed.

Analysis of role-failures is dependent on eliciting appropriate data from user-based evaluation sessions. User accounts of the influences on their decision making, and their interpretation of system appearance and behaviour, are crucial data. This practical necessity is addressed in the following chapter.

5.7. Chapter Summary

This chapter has moved towards the utilisation of models of action in practical evaluation. The first sections describe the useful distinctions that can be made between processing levels, and the different knowledge-spaces that users employ. It is shown that different knowledge-spaces may be used in location of features, recognition and specification. Therefore, it is useful to treat them as separate mental acts, requiring

distinct system support. Each mental act is a potential source of an error phenotype. The original premise from the Chapter 3 model is that these mental acts are system-led. Therefore, user mental acts imply dialogue support requirements. Role failures are therefore described as failures to support a particular mental act.

An approach to analysing protocol data using dialogue roles is described. Protocol data can yield sufficient evidence to account for mental acts, the knowledge-space used, and the aspect of the interface that prompted it. The following Chapter describes a further empirical study in which the model's description of error types is examined.

Chapter 6 - Model-Based Study of Word Processor Users

6.1. Study Objectives

The study had two main objectives. These are

• Testing the power of the model of errors in analysing errors: The model provides a description of how errors occur, characterising user's problems as mismatches between user and system. The study collected a sample of errors from a different type of package to the first study. Given that the model was developed from the study of a draw package, its generalisability as a descriptor of interaction problems required validation on a contrasting type of DM package.

• Study of comparable evaluation methods' performance: This independent study based on the sessions is reported in Chapter 8. The ten sessions reported here were also used to test the MMA method's performance in detecting and diagnosing errors (the MMA method is described in Chapter 7. A further ten sessions were performed, and used to test a (contrasting) method, the Usability Checklist evaluation method (Ravden and Johnson 1989).

Along with the two major objectives, the effects of previous word processor experience on user performance was studied. The Chapter 4 study selected subjects who had little or no experience on the type of package that they were using. The next study involved users who had experience of other packages for the same task domain as the evaluated package. The intention was to observe examples of the transfer problem, and assess if the model of errors can be effective in this context, with or without modification.

6.2. Methods

6.2.1. Word Version 5.1 for the Macintosh

Word 5.1 is a word processing package which uses direct manipulation interaction and is rich in functionality. The editing facilities involve direct selecting and pointing actions. These actions are augmented with a number of functions which can be

activated using menu or icon selection options. The package does not have a standard help facility, but an optional package can be loaded. It was decided that the help package would not be loaded to ensure that the basic system design was tested. The iconic 'toolbar' option was loaded to test alternative cues.

6.2.2. User Subject Group

Ten user subjects, all undergraduate students, took part in the study. All the subjects had some experience of using word processors, but none had used any version of the Word package prior to the session. The subjects had only used WordStar, the standard package used in the University department from which they were recruited, or WordPerfect.

6.2.3. The Task

As with the first study, subjects were given an explanation of the basic functions of the package, and ten minutes to explore the system. They were then asked to perform some basic text-editing tasks using the package. They were given two 'task sheets', one showing the current state of a document, the other showing the intended end state. They were asked to transform the document from its current state to the intended end-state (Figures 6.1.A. and 6.1.B.).

The scenario consisted of a one-page text document which, broadly presented six distinct tasks to the user. These were as follows:

- The centring of the title
- The reformatting of the opening paragraph with bullet points
- The addition of two lines of new text, and an asterisk
- Changing the font of the document, and of individual items to bold and italic
- Moving a block of text
- Reformatting two short paragraph as a pair of columns

The results were analysed using a combination of the observations during the session, and retrospective analysis of the video data.

Technological Innovation

Changes in technology are taking place at a rapid rate in many areas and with these changes an increase in productivity is expected. In many cases, technological innovation has resulted in increased labour output, a reduction in costs and a reduction in the price of the product or service. In many cases high technology requires an increase in capital investment and an increase in employees' skills. The operator often becomes a manager directing and controlling complex machines and processes. With the coming of the computer, which is often an integral part of the machine, the operator may only need to take action when unusual conditions occurs or when a changeover in the product or process takes place.

Well-known examples are the dial telephone, which reduces the need for the telephone operator; hybrid seeds, fertiliser, and farm machinery. Electronic technology has produced phenomenal gains. Since the I950s the speed of computer computations has increased 10,000 times.

Individual Productivity
The productivity of the individual employee can be measured in a different way. If 100 employees produce 3000 units of a single product in one day, the average output might be stated as units per person per day. In this analysis we have no record of a crucial factor, that of time study. For example, the standard time taken to assemble a bench grinder is 2.00 minutes per unit and if the operator assembles 275 grinders during the day, the output is 500 standard minutes (275 x 2.00 = 550). If the operator works an 8-hour day or 480 minutes, the input would be 480 minutes. The operator's performance index would be 114.6 percent.

Staff Originated in the 1980s May undertake broad spectrum of work Report to line managers and supervisors Widely used

Line Introduced in the 1930s and 1940s Growing rapidly Study their own problems Recognition given for outstanding achievements

The average productivity index of a department or of a plant would be the total standard minutes or standard hours produced by all employees divided by the actual minutes or hours worked multiplied by 100. This assumes that all of the operations are covered by time standards. Thus a performance index can be used company wide as a labour productivity index. This form of performance measurement has been used regularly for many years and its use is growing.

Figure 6.1.A. The Task Start-State Presented to Subjects

Technological Innovation

Changes in technology are taking place at a rapid rate in many areas and with these changes an increase in productivity is expected. In many cases, technological innovation has resulted in

- increased labour output
- a reduction in costs
- a reduction in the price of the product or service.

In many cases high technology requires an increase in capital investment and an increase in employees' skills. The operator often becomes a manager directing and controlling complex machines and processes. With the coming of the computer, which is often an integral part of the machine, the operator may only need to take action when unusual conditions occurs or when a changeover in the product or process takes place.

Well-known examples are the dial telephone, which reduces the need for the telephone operator; hybrid seeds, fertiliser, and farm machinery. Electronic technology has produced phenomenal gains. **Note:** Since the I950s the speed of computer computations has increased **10,000** times. The price of the hand-held calculator has been reduced from **$1500** to less than **$10** in a *single* decade.

Individual Productivity

The average productivity index of a department or of a plant would be the total standard minutes or standard hours produced by all employees divided by the actual minutes or hours worked multiplied by 100@. This assumes that all of the operations are covered by time standards.

Staff	Line
Originated in the 1980s	Introduced in the 1930s and 1940s
Widely used	Growing rapidly
May undertake broad spectrum of work	Study their own problems
Report to line managers and supervisors	Recognition given for outstanding achievements

Figure 6.1.B. The Task End-State Presented to Subjects

They were given thirty minutes in which to perform the task. They were asked to provide a continuous verbal commentary on their thoughts and actions, following the protocol analysis procedure of Ericsson and Simon (1984). The instruction sheet asked the subjects to ensure that they reported incidents when they were unsure what to do, had difficulty with an action or had difficulties interpreting the result of an action. A second 'evaluator' subject was briefed to prompt explanations and clarifications of difficulties observed, or expressed by the user. The technique, like the study in Chapter 4, was similar to the 'York Manual' approach (Wright and Monk 1989). Unlike the Chapter 4 study, however, the evaluator was asked to intervene at critical points during the session. This alteration to the approach was made in order to accommodate the testing of an evaluation strategy, which is reported in Chapter 8.

The subjects were asked to provide a continuous verbal commentary, describing their thoughts and actions. They were accompanied by an evaluator, who was briefed to ask for clarification of critical incidents, during the session (see chapter 8). The intervention technique was similar to that described in Wright and Monk (1989). The sessions were video-recorded and user verbalisations transcribed. As in the Chapter 4 study, the subjects were asked for further descriptions of the errors observed during the sessions. This was done first by the accompanying evaluator, and separately by an independent observer.

6.3. Data Analysis

6.3.1. Error Phenotypes

The observed errors were classified according to the surface characteristics of the error incident. The analysis used and extended the concept of error phenotypes described by Hollnagel (1993). The original definition of phenotypes is overt observed behaviour of the type described by Wright and Monk (1989). The extended definition of phenotypes uses verbal protocol evidence. The phenotype categories broadly reflect the sequence of an action, reflecting points in the cycle where users declare problems. The classification was done both to provide a neutral analysis of incidents, and to study the nature of links between phenotypes and genotypes. Four error phenotype classes were observed. They were linked to the sequence described in the model of action. The category 'reject feature' was added to describe occasions when users aborted after inspecting a feature. Unlike the error categories used in the Chapter 4 study, they simply describe phenotypes rather than an attempt at diagnosis. The categories are linked to stages in the cycle of action (Fig 6.2.) and listed below:

Observed user Activity	Ideal Result	Associated Error Phenotype
Search	Find candidate feature	Fail to find feature
Explore	Proceed to task action	Reject feature
Try Action	Perform Manipulation	Accident/ manipulation difficulty
Evaluate Result	Satisfactory/ proceed	Unsatisfactory result

Figure 6.2: Error Phenotypes Linked to Stages of an Action

Failure to Find Features: Subjects declared a goal and scanned the interface for features. Instances of subjects scanning but giving up without trying a feature.

Rejection of a Feature: Cases of subjects declaring a goal, selecting and inspecting a feature (declaring experimentation), then rejecting it as irrelevant to the goal.

Unsatisfactory Result: Cases where a subject performs a task-action, but expresses dissatisfaction with the result.

Accidents/Manipulation Problems: Difficulties with manipulations, or declared 'slips' of the type described by Reason (1986).

6.3.2. Genotype Analysis

The error phenotypes were analysed and placed in the most appropriate genotype category. The Genotype Categories used were the set of roles failures described in Chapter 5. Also, and extra category 'non-error' was created. These are cases where the error does not appear to imply the need for a design change. Some user errors may merely be examples of the subjects learning about the interface.

6.4. User errors made in the Sessions

6.4.1. Overview

No subject completed the whole task within thirty minutes. Subject 8 thought the task was complete with two minutes to spare, but had left out the task of centring the title. The ten subjects made a total of eighty-seven errors between them. There were some individual differences in performance, although errors were fairly evenly spread amongst subjects (see Figure 6.3.).

Task \ Subjects	1	2	3	4	5	6	7	8	9	10	Total
centre	1	5		1			1	1	1	1	11
bullet	2	2	1	3	2	2	2	2	3	2	21
move	2	2	1			2	2		1	3	13
change	2		2	3		1	1		2		11
columns	1	3	2		4	1	2	3	3	3	22
scan				1							1
enter		1		1		1				1	4
repair					3			1			4
total	8	13	6	9	9	7	8	7	10	10	87

Figure 6.3: Errors by task for each user subject

The errors were first classified by the task that the subject was attempting when an error was made (Figure 6.3.). The column task (22 errors) and the bullet task (21 errors) proved to be the most consistently troublesome. The columnising task involved six cases of subjects selecting text, and then the Column icon on the tool bar. The necessary step of creating a page-break was missed by these subjects. There were also five cases of subjects failing to find appropriate facilities for performing the columnising task. The other errors were a consequence of subjects' inability to find a way of using the columns feature satisfactorily. Another three subjects experienced difficulty aligning the text into columns manually. The word-wrap feature made it difficult for them to align columns.

The bullet task caused a similar range of problems. The system provides both a bullet icon and a menu option. However, five subjects failed to find the features. There were also five subjects who had difficulty with the placement and indentation of bullets. The feature makes an automatic indent when the placement command is made by the user.

6.4.2. Error Phenotypes

The observed errors were assigned into Phenotype categories as described in section 6.3.1. above. Figure 6.4. shows the totals for each section. The totals show the most common phenotype (52%) to be an unsatisfactory result of a task-action. All of the ten subjects made between three and six errors of this type. Next most common problem was inability to find features, which accounted for 24% of all errors made. Rejected features accounted for 15% of all errors. There were eight examples of accidents and manipulation difficulties (11% of the total). The more frequent errors are shown in Figure 6.5. Some phenotypes in the Accident/Manipulation and Unsatisfactory Result categories refer to operations rather than specific features. These are cases where the users' reports of the error refer to a problem with a general operation rather than an individual feature (e.g. difficulty operating menus).

Phenotype	1	2	3	4	5	6	7	8	9	10	Total
Unable to find features	3	3	2	1	2	1	2	1	3	3	21
Rejected Feature	2	1	0	1	3	1	2	1	1	1	13
Accidents/ manipulations	0	5	0	1	0	1	0	1	0	0	8
Unsatisfactory result	3	4	4	6	4	4	4	4	6	6	45
Totals	8	13	6	9	9	7	8	7	10	10	87

Figure 6.4: Error phenotypes observed in the ten sessions

Error phenotype	No. of subs	No. of instances	phenotype category	Error phenotype	No. of subs	No. of instances	phenotype category
Menu failure to operate after command	8	11	UR	Change Case icon	3	3	FF
Centre	7	7	FF	Auto-text position/ return	3	3	UR
Column icon	6	6	UR	Creation of gap for pasted text	2	2	UR
Bullets	5	5	FF	Paragraph menu	2	2	FF
Select and move by carriage return	5	5	UR	Draw Package icon	2	2	FF
Bullets (auto indent/placement)	4	5	UR	Page Break	2	2	FF
Columns	4	4	FF	Accidental option selection when scanning menu	2	2	MAN
Double icon select	4	4	UR	Bold *menu*	2	2	UR

FF = Failure to find features

UR = Unsatisfactory result

RF = Rejected feature

MAN = Manipulation Difficulties

Figure 6.5: The more frequent error phenotypes observed in the sessions

6.4.3. Failure to Find Features

The twenty-one cases of subjects failing to find features were mainly confined to three tasks. Seven subjects were unable to find facilities for moving the title of the document to the centre. Centring can be performed by selecting the appropriate text, and adjusting the settings on a ruler bar which is located immediately above the document. All users were observed searching the menus and the icon bar for the feature. Five subjects failed to find facilities for adding bullet points to the text. Word 5.1. has a facility for selecting and placing bullets. This is operable both from the menu and the icon bar. The menu option is called 'bullet'. The icon has three square dots accompanied by horizontal lines. There is also a symbol menu bearing an appropriate option (which can be operated in a similar way). The icon is visible on the screen.

Four subjects failed to find a suitable feature for creating columns. This included subject 3 who, having tried and failed to use the columns icon, searched unsuccessfully for an alternative feature. Two subjects described facilities for selecting and moving text, but did not find the cut and paste facilities.

6.4.4. Rejected Features

The thirteen errors in this category were spread amongst eight features. The 'change case' icon was erroneously selected on three occasions. This is represented by three images of the letter 'B'. This was misinterpreted by two subjects as representing the 'bullet' feature. Another subject thought that it represented the 'bold' feature. The 'paragraph' menu was erroneously selected by two subjects. Subject one was performing the bullet task, and Subject 2 was looking to move text. The draw icon, which brings up a draw package similar to MacDraw, was also selected by two subjects. Subject five expected it to assist with column creation. Subject six expected it to help with the bullet addition task.

6.4.5. Accidents/Manipulation difficulties

Subject 2 experienced some considerable difficulty with text select feature (4 errors). Three of the other four errors in this category involved difficulty in controlling pull-down menus. Subject 6 accidentally selected the undo feature while searching for pasting facilities. This resulted in a further error when she subsequently pasted the wrong text.

6.4.6. Unsatisfactory Result

There were eleven examples of subjects finding that menu options did not respond. Ten of these involved subjects failing to select a target item before selecting a menu item or icon. For example, four subjects tried to change the whole of the text font by selecting the 'Times' menu option, without selecting any text.

Six error incidents involved subjects attempting the wrong procedure for using the column icon. The six subjects who used the feature all selected text first, and then the icon. They expected the selection of the icon to turn the selected text into two columns. None of them discovered the correct procedure for the task. None of them were aware that the procedure involved isolating the text with a page break.

The unintentional deletion of text using select and return occurred on five occasions. The subjects selected a section of the text, and then pressed carriage return on the keyboard. The selected text disappeared unexpectedly, leaving the subjects with either a retrieval or retyping task. All subjects who made this error admitted that they expected the text to be moved as a block rather than be removed.

Four errors in this category involved the 'cut and paste' facilities. Subjects were asked to move a paragraph to a new position. Two subjects created a gap for text to be pasted into. The gap that they created remained after the paste was performed. One subject performed a cut, by selecting the target text, moving the cursor to the desired location, and selecting 'cut' from the menu. The subject was not aware that the paste facility was needed to complete the action. A further subject pasted the wrong text after an inadvertent undo.

There were four instances of subjects expressing dissatisfaction with the behaviour of the bullet feature. The feature automatically places a bullet in a selected location, automatically indenting the text. Four subjects complained that the feature did not afford sufficient user control. Three subjects had problems with the word-wrap feature. All three expressed frustration with the feature when trying to manually arrange text in columns. Two subjects made errors with the 'bold' menu feature. The subjects altered an asterisk to bold type, continuing to enter text next to it. The subjects expressed annoyance when the feature continued to produce bold text.

6.5. From Phenotypes To Genotypes

6.5.1. Overview

The phenotypes were further analysed to investigate three points of interest. These were:

• The distance between Phenotype and Genotype
The descriptions demonstrate the amount of analysis required to establish a genotype from the surface characteristics of an incident.

• Criteria for distinguishing a system error from a user error (including error severity)
Not every user problem may be considered a system error. Criteria for establishing the system's failure include the number of subjects who made the error against the number who tried and succeeded. Also, the effects of an error are taken into account. The error study in Chapter 4 did not account for the phenomenon of learning through trial and error. However, this has been cited by Shneiderman (1987), Sutcliffe (1988) and others as a desirable feature of DM interfaces. If an error is made by a user, but that user is able to recover and perform the action satisfactorily, this serves as evidence against there being a design problem.

• Causal connections between individual errors
Some incidents revealed more than one problem with the design (e.g. a primary and secondary cause). Also, some user errors are the remote effect of a previous system error. It is crucial that the diagnosis phase deals with this.

6.5.2. Analysis and Categorisation

The protocol data collected often required a clarificatory analysis with the user.

For example, the evidence for feature identify problems was very similar to locators, with further investigation frequently necessary. The analysis used evidence from the session, and retrospective questioning involving 'constrained search' tests. The Constrained Search tests were tried in cases where subjects had been unable to find relevant features. The tests involved pointing to a screen area such as a menu or icon array, and asking the user 'which one of these is the necessary feature for this task' (pointing to the task sheets). Feature identification was deemed the likely problem if they still failed to identify. Otherwise, the problem was deemed to be the lack of visual assistance in constraining search.

Another technique used was to ask subjects why they had chosen a subsequently rejected feature ('motivation analysis'). This served as a prompt revealing either a misapprehension of screen layout, or the absence of spatial information to help search and identification. This was done to establish whether the interface had prompted an incorrect choice, or simply failed to provide any assistance in search and feature selection.

The goal-tree analysis used in the Chapter 4 study was used for some of the 'unsatisfactory result' and 'Manipulation' phenotypes. The subjects were first asked to describe their expectations of how a task could be performed. The sequence described was then compared to the actual task-action sequence on the device. The analysis examined cueing and presentation issues, along with task structure. The user's account of prior expectations and system model representation were compared both for resemblance as task structure, and for naming and description of steps. This elicited the influence of prior system knowledge or domain knowledge. In doing so it captured the reasoning behind erroneous assumptions. The analysis also reveals cases (such as Subject H using the palette default to make arrows in Chapter 4) where an inappropriate feature is selected but not rejected by the subject.

A supplementary analysis similar to that used in section 4.3.3. was conducted for selected 'unsatisfactory result' errors. Subjects who made the relevant errors were asked to perform a manual equivalent of the device-task. This analysis was targeted towards errors where the user had expressed an incorrect model of the task, revealing abstraction-based generalisation as a source. This revealed the difference between the user's device independent task-model, the task-model that the user expressed during the session, and the system model. In particular, this was used to study problems using the 'column' icon (see Section 6.5.5.).

The phenotype 'accident/manipulation difficulty' was counted as an execution problem only if further study of the procedures employed and the visual feedback failed to reveal problems.

Subjects were prompted for their beliefs about the initial system state when an action commenced ('system state analysis'). Subjects comments about failed action may reveal a misinterpretation of the system state prior to the action. The incident was therefore linked to the action where the relevant system state change occurred. The videotape of the incident was then re-run and the user asked if they were aware that the change had occurred. If the user was unaware that the change had occurred at all, the error is classified as a Perceive Feedback problem. If the user is prompted to act by a misinterpretation of a state-change, this was deemed an Understand Feedback

problem.

The following sections describe the incidents that were assigned to each category. The more frequent examples are summarised in figure 6.4. above. Incident types (e.g. four cases of failure to find columns = one example of a type) were assigned to genotype categories as a result of the analyses described above. The 'incident type' refers to observed incidents over the ten sessions. For example one incident may have been observed in six sessions. A description of the techniques employed to establish genotypes are provided alongside the incident types that were analysed.

6.5.3. Locators

There were six incident types where the clear primary was locator problems. Three of these were 'unable to find feature' phenotypes, and three were 'reject feature' phenotypes. The seven cases of subjects failing to find centring facilities were deemed to be locator problems. All the subjects who had this problem were observed searching menu and icon features, without, apparently, noticing the ruler bar. The subjects later had the ruler bar pointed out to them. Five said they did not know it was there, and the other two said they did, but did not realise what it represented.

The four failures to find column making facilities revealed Locator problems. The four subjects who did not find the column icon were given a 'constrained search' test. This involved the independent observer pointing to the appropriate icon bar, and telling the user that one was the Column icon. All guessed successfully, suggesting that the design of the actual icon was adequate.

Locator problems also emerged from failed actions. In these cases, the root problem was misguided search. Two examples of subjects selecting and rejecting the Draw package icon were classed as Locator problems. Both subjects were asked to provide a rationale for their selection. One subject was looking for an alternative way of making columns after failing to use the Column icon. He admitted that the choice was largely random, although the shapes on the cue suggested that the feature may provide for reconfiguration of text blocks. The other subject was looking for the 'bullet' facilities. This subject also admitted that the search was not directed, but saw the cue and thought it may offer symbols to add to text. Subject 8 tried the justify icon for columns, and Subject 5 the 'New Document' icon, and gave a similar explanation.

6.5.4. Feature Identifiers

There were six incident types which were classified as Feature Identifier problems.

Two of these were from 'failure to find feature' phenotypes, three from 'reject feature' phenotypes, and one from an 'unsatisfactory result' phenotype. Six subjects failed to find Bullet creating facilities despite scanning over both the icon and the menu option named 'bullet'. All the six subjects claimed that 'bullet' was not a familiar term. The icon was recognised by four subjects in a 'constrained search' test. Also, three subjects had recognised and used the bullet icon. This suggested that the icon was more effective than the menu option name, though still not ideal. The icon was also rejected by two subjects. Another subject recognised the feature, but rejected it because the icon shows square dots, whereas the task-sheet showed round bullets (the feature produces round bullets).

Three subjects failed to find facilities for moving text. All three claimed that they had been looking for a function called 'move'. They were directed to the appropriate menu and asked if any feature on it could be the one. One subject suggested that 'copy' may be the appropriate option, but the other two felt unable to offer a suggestion. One subject failed to remove the symbol window after calling it. The window is removed using a standard Macintosh icon, the small box in the top-left corner of the window. The subject did not know about this feature when asked.

Three subjects selected the change-case icon for the wrong task-action. Two subjects (who knew the term 'bullet') thought the feature would give them bullets, and one thought the text would change to bold. The two who expected the 'bullet' feature assumed that letter B in the icon represented it. The other subject pointed out that the front B looks like it is in bold type, strongly suggesting relevance to changing the type to bold. The examples suggested the feature was ambiguously cued, and therefore a 'feature identifier' problem.

Two subjects selected the 'paragraph' menu option for the 'bullet' task. One said that he expected to have to specify the dimensions of the amended format, and that 'paragraph' seemed a likely feature for doing this. The other simply claimed that there was little indication of what to do, and he had scanned and selected 'paragraph' in speculation. This subject was shown the correct procedure and asked if this operation had been contemplated. He responded by admitting that he was not expecting such a straightforward procedure. This was classified here as it was felt that the 'paragraph' feature had 'drawn in' the subjects and created bogus expectations. Another subject used 'copy' rather than 'cut', for the text moving task. This suggested that the terms for similar but distinct operations may be sub-optimal. The subject seemed ignorant of the cut/paste metaphor, a common Macintosh feature.

Five subjects had selected a block of text followed by the return key. This removed

the text from the screen. Had the text not been removed, this may not have been sufficiently severe to be called a system error. The subjects could have learnt that the procedure was wrong without major difficulties. The problem was seen as the discrepancy between the 'return' label on the key, and the feature's behaviour. The subjects were given no indication that it worked as a delete under certain circumstances.

6.5.5. Specify Operation

Nine incident types were classified as 'specify operation' problems. Eight were 'unsatisfactory result' phenotypes and one was a manipulation problem. Six subjects selected the column icon for a selected area of text. When questioned retrospectively they described their prior expectations. This is shown alongside the system model in Figure 6.6. They all felt that the task appeared to be a straightforward example of the 'select text - select feature' procedure used across a range of tasks. Three subjects attempted to reverse an action by pressing an icon a second time. One subject tried to undo selection of the 'new document icon' by reselecting the icon and two tried the same operation to reverse bullet placement actions. Conversely, Subject 3 undid a completed font changing task by pressing an option twice. He admitted later that he was trying to continue, whereas the others admitted trying to reverse an action. All four subjects stated that they had expected all the icons to operate in the same way, having generalised a rule from previous option selection.

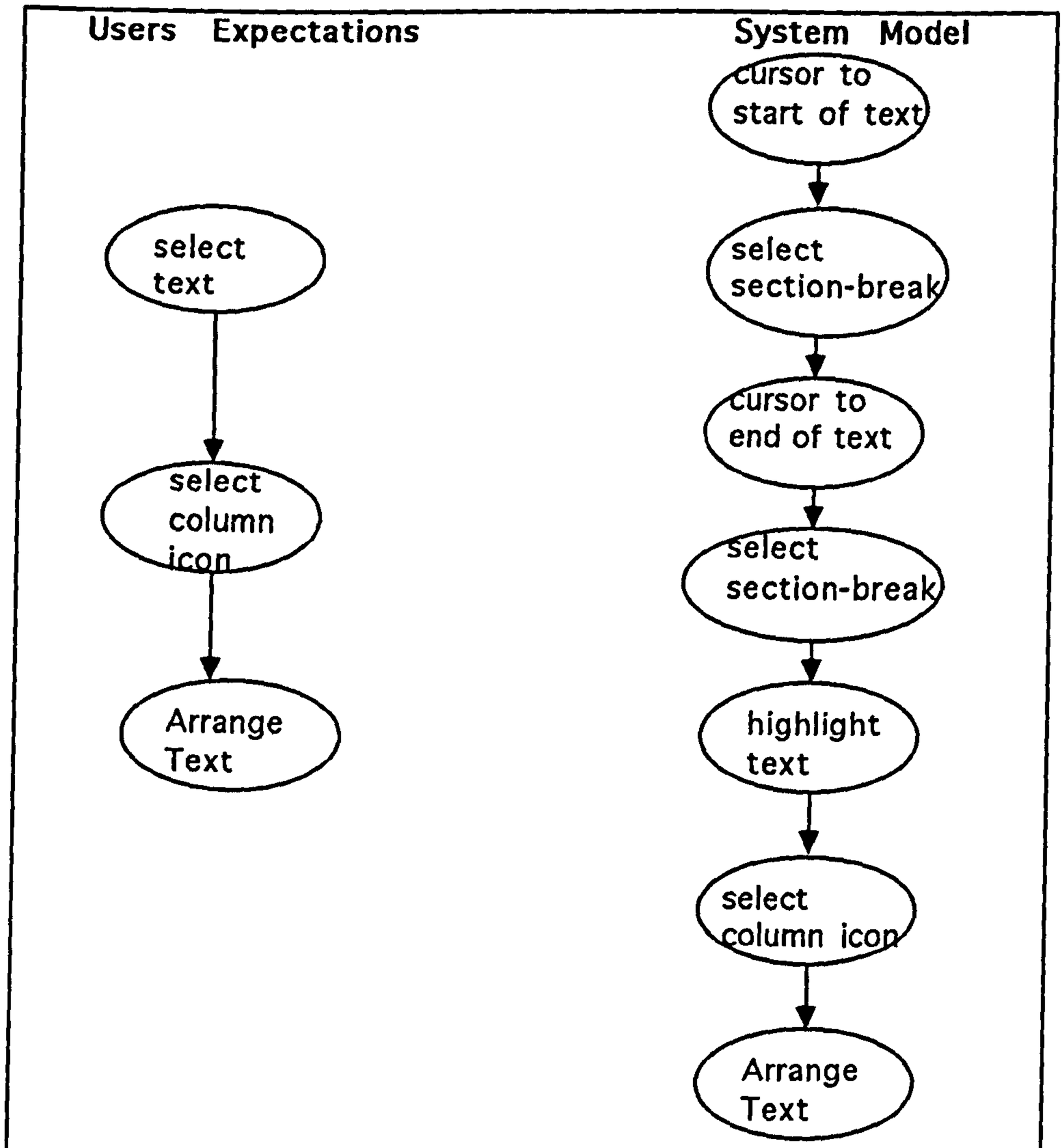


Figure 6.6: User expectations contrasted with the system model for the columnising task

Users' models of the columnising task were further investigated. A goal-tree analysis was performed, similar to those conducted in the Chapter 4 study. Two subjects were asked to specify their device independent task-model. They were asked to describe how they would manually edit a paper mock-up of the task. Their descriptions both involved a sequence which was closer to the system model than to their expectations of device behaviour, declared during the session. This is shown in Figure 6.7. The system model reflected the order of steps in the device-independent models. However, the system model did not match the expectations the users held from their knowledge of previously selected icons and menu options.

The column icon suggests that use of metaphor in design may interfere with user learning. As discussed in Chapter 5, metaphor may suggest operational rules, where the user does not know the rules. Also, metaphor may assist the selection of rules where it is unclear which rules the user should apply. However, the subjects who tried the column action were prompted by the system to apply known operational rules for using icon operations on specific text areas. The fact that the sequence of operations resembled the device-independent task rather than 'standard' device operations made the column feature baffling to users.

Two subjects did not take the necessary steps to perform cut and paste for moving text. They described their surprise at no menu being produced when they performed their actions (one simply selected 'paste', the other 'cut'). Both expected a system response, although neither were specific about the expected response. It was classified as a 'specify operation' problem on the grounds that the 'cut and paste' metaphor appeared to be ineffective. The procedure contains two menu selections, unlike the majority of operations. Therefore, the user must comprehend the link between the concept 'cut' and the concept 'paste' in order to work out the procedure. These subjects had failed to do so. Both claimed that they had expected system feedback indicating further steps.

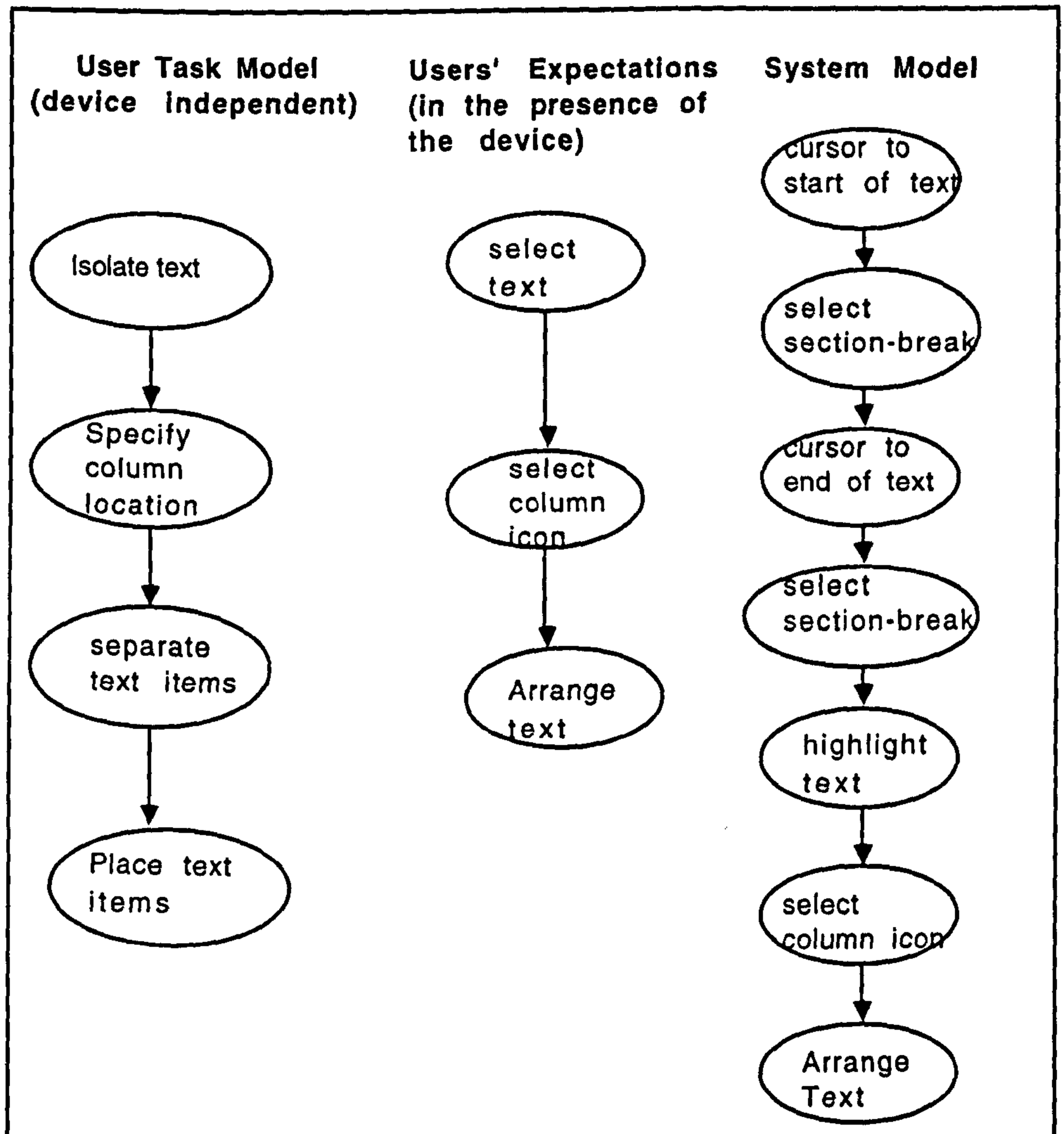


Figure 6.7. Two subjects' task models for the columnising task, alongside their device expectations and the system model

Five subjects had difficulties with bullet placement. Again a goal-tree analysis was performed for each subject. The subjects had anticipated having to create margins for bullets. Also, the placement was not accurate enough for two subjects, who found that the sequence [select paragraph-select bullet] resulted in a bullet placed one line above the paragraph. All five subjects found this action difficult to edit or reverse. Four of the subjects attempted manual editing as an initial response, therefore removing the option of using the 'undo' feature. A goal-tree analysis revealed a significant contrast with other examples, namely a user model more detailed than the system model. This was the result of the subjects expecting, and wishing, to perform subsequent editing on the bulleted paragraphs. In other words, the feature automated too much, and did not allow enough user control.

Two subjects had difficulties with the procedure for changing from plain text to bold. The task involved adding a symbol in bold, with plain text entered next to it. Afterwards the subjects said that they expected the change to bold to operate only on the selected area. However, the change affected subsequent text, despite the fact that both subjects had entered the item in plain text.

Two subjects had problems with cursor placement. They tried to move the cursor to the right-hand side of the screen. The system did not allow them to plot freely with the cursor. The cursor cannot be placed directly on areas to the right of a carriage return. This was counted as an operation specifier problem. The subjects' model of the page metaphor had no such restriction, and task-action was specified on the basis of this assumption.

Three subjects had difficulty holding and scanning pull-down menus. Further analysis revealed that this was a result of them expecting the menus to remain visible until deselected. One subject accidentally selected 'undo' without realising, which undid a cutting action. The others accidentally selected redundant options (that did not work without 'select'). This occurred because the subjects released the mouse button, believing the menu would remain.

6.5.6. Execution

The study found no clear cases of execution failures, although there were eight recorded 'manipulation' phenotypes. However, all manipulation phenotypes were primarily classified as either non-role failures, or Specify Operation problems.

6.5.7. Perceive Feedback

One 'unsatisfactory result' phenotype was placed in this category. Subject 10, believing the selection of the 'new document' icon to have erased the text, searched the tool bars for facilities to reverse the action. The selection of the icon had brought up a new document over the existing one. The screen gave the appearance of a total deletion, apart from the document header at the top of the screen. This was not noticed by the subject, who panicked believing that the text had been erased.

Subject six pasted the wrong text after inadvertently undoing a cutting action. Retrospective questioning revealed that she was unaware that she had previously altered the clipboard status accidentally.

6.5.8. Understand Feedback

One 'unsatisfactory result' phenotype was classified as an understand feedback problem. Subject eight had opened and then closed the draw package (which appears as a sub-window), and was confused by a large cursor which remained on the text document after the draw package closed. The subject balked at this, and paused for a substantial time. He eventually tried to remove the cursor by selecting it as an area, and pressing the delete key.

6.5.9. Non-Role Failures

Eight error types were not considered to be problems with the design. In some cases the design had caused difficulty, but independent analysis suggested that the design could not be usefully changed. Nine of these were from 'unsatisfactory result' phenotypes, one from 'reject features', and one manipulation difficulty.

The ten cases of failing to select text before an action were spread over eight subjects. No subject made the error more than twice and all eight subjects subsequently performed successfully. The alteration of the font from Helvetica to Times could have been performed with the 'select all' feature, but the subjects performed a manual select. None of them searched for a 'select all' feature.

Two subjects inserted a page-break as the first action in the Bullet placement task. Both said afterwards that they expected to have to insert a break in order to select the relevant text block. One subject realised soon that the manual area select could do the job. The other remained confused about what to do, abandoning the task. However, this subject learned the manual select and applied it (albeit with some difficulty) for

other tasks. Both errors were made early in the session and were recovered from.

Two subjects created a gap in the text before pasting cut text. When asked they claimed that they thought the gap was needed for the text to fit. Both revealed that they had later understood the principle of operation. Both saw the after-effect of the cut. The independent analysis concluded that the feedback was comprehensible and generalisable. Therefore, the error could be described as system learning.

One subject mistook the 'ghost' cursor for the actual cursor and typed in the wrong place. This was not considered a design problem. The only alternative was to have the cursor move with the mouse at all times. This was considered likely to cause greater problems. Another subject tried to select two areas of text consecutively, an action that the system does not allow. The system showed clearly what the restriction on selecting action was (i.e. select and operate on single areas).

One subject selected the wrong item from the symbol menu. However, the subject immediately saw the problem returned and rectified the error. Further examination of the menu by the independent analysis failed to observe a problem with the cueing. The error was assumed to be a loss of concentration by the subject.

6.6. Secondary Causes

6.6.1. Introduction

The problems described above were assigned to their primary genotype causes. However, some errors appeared to provide evidence of more than one design flaw. This phenomenon is now discussed using examples from the sessions.

6.6.2. Locators/Feature Identifiers

Locators, along with aiding navigation, provide contextual information which assists feature identification. The evaluator is faced with task of deciding if there is a problem with one or both of the roles, from similar phenotypes. Therefore, poor locating information may contribute to a 'feature identifier' failure. The problem that subjects had with the change-case icon demonstrates this point. The change-case icon shows two large letter 'B's and another in lower case (which is smaller and relatively hard to see). This was mistaken for the 'Bold' feature, and for the 'Bullet' feature. Subjects who made this misinterpretation associated the 'B' with their tasks. In the 'bold' example, this was compounded by the fact that the letter 'B's on the icon look like they

are in bold type. The cue design is clearly a problem. However, this is compounded by the absence of a group identifier. Had the partitioning of feature types been clearer, the subjects would have had a further clue as to the feature's identity (e.g. a clearly labelled icon cluster). This example demonstrates that a single phenotype may provide evidence both of a general problem with locating features, and poor cueing of a particular feature.

6.6.3. Feature Identifiers/Operation Specifiers

An example of this dual classification was two sets of error genotypes referring to a single feature. For example, the cut and paste facilities spawned 'Feature Identifier' and 'Operation Specifier' genotypes. The evidence of design problems from these two examples appears to be complementary. The subjects who failed to recognise the feature were looking for a 'move' function. Similarly, the subjects who selected a single function ('cut' or 'paste') had not expected the procedure to be as it was, with two option selections. They had not comprehended the cut-paste metaphor.

6.6.4. Operation Specifiers/Feedback

The column-making feature was described as a problem with the specification of operations. However, the feature shows a secondary problem of feedback support. When the icon is selected, the whole text becomes one narrow column on the left-hand side of the document. Therefore the user has caused a state-change, but one that is neither desired nor comprehensible. The feedback failed to inform the user about the current state of the task or the correct action that would have been required.

6.6.5. Conclusions from Genotype Analysis

The phenotype/genotype distance may be wider than is apparent during the sessions. Examination of incorrect feature choices or failures to find features may indicate high-level or feature specific problems. The true nature of these problems may only be established using retrospective questioning and model elicitation. A number of these errors were classified using analytic techniques such as 'constrained search' tests. The goal-tree analysis revealed the lack of detailed expectations in a number of cases. Subjects seemed to anticipate that device feedback would indicate further actions. The subjects showed a notable reliance on menu or icon led specification.

Error frequency amongst subjects is potentially misleading statistical evidence of a system error. The most frequent recorded error over the ten sessions was failure to select text prior to an operation. However, this was not seen a system error, because

subjects appeared to learn. A simple statistical analysis would wrongly suggest an important design problem.

Error sequences, such as the 'new document' selection problem, are self-evidently clear from the phenotypes. The 'new document' example shows first an incorrect feature selection (the icon) then two incorrect procedures (reverse by reselecting the icon, use of document search facilities). Three error phenotypes are recorded from this sequence. Concurrent protocols also reveal that the two incorrect procedures imply a feedback problem (namely, that the text appeared to have been deleted). The evidence presents a number of alternatives for redesign. These include improving the cue, making the feedback clearer, and standardising the effects of a second selection of the icon.

More distanced error pairings (e.g. accidental menu selection--pasting wrong text) may be established by tracing the subject's view of the interaction 'history'. Such a view can be gleaned from references to the state of interaction prior to the latter of the two errors. In the 'wrong text' example the subject revealed ignorance of the previous accidental selection. The space of design issues includes questions such as 'is the menu manipulation awkward?', 'is the procedure optimal?', and 'is the feedback on the clipboard status adequate?'. The evaluator is left to consider redesign options based on these findings.

The help given to the evaluator on the nature of redesign varies with the type of error discovered. Subjects usually declared a name of the feature they were searching for, hinting at an optimal name for textual cues. This information may contribute to redesign. It may harder to suggest the redesign of icons using formative advice from evaluation sessions. Subjects' accounts of 'specify operation' mismatches give an indication of alternative designs. However, two problems emerge. One is that subjects may not have well-formed expectations in some cases. Also, a sample of 6-10 subjects may not have models that concur.

6.7. Analysis of Interference from Previous Package Use

6.6.1. Introduction

A further study was conducted to examine the influence of previously used word processors on the performance of subjects. References made by subjects, either concurrently or when cross-examined, to packages used before, were isolated and analysed. These were used to establish clues to interference effects. Subjects had

experience with broadly two word processors, namely WordPerfect and WordStar. It was found that a total of ten error types were probably influenced by interference effects, including two that had been classified 'non-role errors'. These are now examined for their effects on subjects' behaviour at various points in the cycle of action.

6.7.2. Examples of Errors

The subjects who failed to find cut and paste facilities admitted that they were searching for a feature called 'move'. They were both subjects who had substantial experience of using WordPerfect prior to the session. The equivalent task in WordPerfect can be performed using a command called 'move'. The subjects responses suggested that the strong association between the task and the term 'move' had made the problem harder. Other subjects, including three who had not used WordPerfect did find the features. This suggests that naming, and the association of particular names to tasks may interfere with feature search.

The problems with 'cut and paste' procedures experienced by subjects who were WordPerfect users suggest that expectations of task-structure may be influenced also. The two subjects selected one command on the menu, and were then lost for what to do next. Both said that they had expected more to happen as a result of their actions. The WordPerfect 'move' function was cited by one of the subjects (this feature of WordPerfect performs text moving tasks with fewer user operations than 'cut and paste')

Three recorded problems with the performance of menu pulling and scanning were deemed to be transfer-influenced. The subjects had all used the menus for scanning and selection prior to the incidents. However, they had tried to hold the menus by going to the top option and depressing the mouse-button. This is the correct procedure on Microsoft Windows. On the Macintosh it results in the top option being selected, rather than the menu remaining in view. Subject six selected 'undo' in this way, and subsequently pasted the wrong text. This shows that internalised procedural skills may interfere in user performance on similar, but operationally different, devices. This is a problem similar to pilots or drivers changing from one cockpit design to another, where automated procedures may be inappropriately retrieved.

6.7.3. Other Transfer Evidence

Interviews with the subjects, revealed other possible transfer effects. Those that had incorrectly selected 'page break' or 'paragraph' suggested that they were used to

selection of text and screen areas using commands. This is despite the fact that they had been shown the operations during the training sessions. Four other subjects admitted to initially searching commands rather than utilising mouse operations. They referred to the early period of the session, saying that the absence of familiar options forced them to try unfamiliar manipulations. This suggests that the learning and internalisation of basic procedures may be slightly retarded by interference.

It seems that the WordStar package caused less interference than WordPerfect. This may be because there is a clearer contrast between the style of WordStar interaction and Word. The observed problems that WordPerfect users had tended to be where there was a closer resemblance in interaction style and command names.

6.7.4. Conclusions From Transfer Effect Study

Whilst the study allows only tentative conclusions to be drawn, some points are raised. A command-based word processor may cause relatively little interference by comparison with one that bears a physical and operational resemblance to the target system. This is despite the fact that the more command-based interfaces require the user to learn and retain device knowledge. Most of the observed problems seemed to be misleading memory and action triggers. These seemed largely in areas where familiar tasks had some characteristic which was similar between packages.

6.8. Review of Methods Used in the Sessions

6.8.1. The Constrained Search Test

These tests proved a valuable source of information. It was clear that locating features was a general problem. It would have been difficult, therefore, to assess the merits of individual feature representations without such a test. The tests afforded the chance to separate individual features cue problems from more general problems with the visual layout.

The difficulty with this technique is selecting a suitable area in which to constrain the search. The search was constrained to one or other icon bar in most of the examples. However, in other examples it may be less easy to avoid over or under constraining the search space.

6.8.2. Motivation Analysis

The success of asking users to reason about (erroneous) feature selection was

relatively limited, although it is useful for spotting ambiguities in the naming of features. Often the user offered little information of substance in response to these questions. This suggested that subjects simply speculated in choosing features.

6.8.3. Goal-tree Analysis

The advantage of using this analysis in concurrent and retrospective analysis was clear. The capturing of assumptions made from generalised system knowledge is facilitated, as a number of examples (e.g. Columns) demonstrate. Also, the frequent lack of detailed expectations declared by subjects highlights the system-led nature of the dialogue. In cases where subjects look for the interface to give cues in response to each action, the user will not have well-formed expectations.

6.8.4. System-State Analysis

This analysis gleans useful information about more distant causes of overt errors. These errors are caused by a hidden result of a previous action. The user's declaration of the system state at the point of a critical incident reveals ignorance of a necessary state change. However, the analysis is hard to reliably use retrospectively, unless video playback facilities are available.

6.8.5. Summary of Techniques

Assessment of the techniques' effectiveness should take into account the inherent difficulties in pinpointing causes. The goal-tree analysis successfully indicates the state of user models at critical points. The sources of current decisions can be linked to display-led strategies for action. The boundary between locator and feature identifier mismatches is unclear. It may be difficult to use the information gathered as more than evidence to support inspired guessing about the real design problem. However, the distinction represents a useful 'diagnosis space' for the evaluator to explore. It could be argued that the true genotype in these cases may never be fully established. It is simply a matter of finding an appropriate solution to a design problem.

6.9. Discussion

The use of the role mismatch model to describe errors seems effective in augmenting analysis of user studies. However, the model requires a battery of investigative techniques to establish probable genotypes for design improvements. Evidence for the

nature of improvements to textual feature cues comes from the subjects' citation of what they are looking for. Similarly, misleading iconic representations may be pinpointed by subjects' verbalised interpretations. Failed search requires follow-up analysis to establish whether the design of the feature cue and/or higher level contextual information require attention.

Some of the Operation Specifier examples suggest that users' device independent model of tasks may not significantly influence their expectations (particularly where a partial device rule-base has been established). The 'column' example, illustrated in Figure 6.7., suggests that users understand, even expect that steps in familiar tasks may be ordered differently on the device. This is strong evidence that action on DM systems is situated, and that the 'rule-base' compiled by users may be a stronger influence on behaviour than their device-independent model of the current task (although their device independent model may contribute to the learning of rules).

The claim of the model of action, that procedures are specified only after feature selection, may be further elucidated. The users task-model seems does not seem to create strong expectations about sequences and their ordering. Also, some tasks may simply be unfamiliar to users. The examples appear to suggest that feature types are bound to device operations by users, causing them to anticipate a novel ordering of sub-tasks. The 'column' feature design fails, not by introducing an extra step beyond the user model, but because it makes access to the task too difficult. The subjects searched for a matching between [task-token{make columns}], and [device token {icon-menu option}]. This typical rule-based behaviour was not supported.

6.10. Implications for Evaluation

Empirical findings were analysed further, to establish implications for the theory and practice of usability evaluation.

The analysis shows that DM packages, particularly ones where the locator problem is serious, require deep analysis to pinpoint poor visual cueing. The problems with feature selection usually required application of constrained search tests or questioning on subjects rationale for (mistaken) choices. The study demonstrated that high-level spatial metaphor and feature clustering contributes more than simply navigational aids. Many of the feature representations in Word are potentially ambiguous. Much of the ambiguity seems to be a problem of context. A number of cases show feature representations bearing seemingly 'natural' names or images. However, a number of these features were either not found by users or selected incorrectly, suggesting the

need for redesign.

It may be possible to investigate search and feature identification, to assist preliminary evaluation. Subjects could be shown the interface and asked to identify screen areas, feature groupings and individual features. Furthermore, this could be done using paper-based scenarios or rapid visual prototypes, bringing evaluation forward in the design process.

The analysis of 'operation specifier' problems reveals a more varied set of possible mismatches than hitherto anticipated. The problems with auto-placement and auto-indent for bullets exemplify the danger of over-automation in DM task-design. Subjects found the feature too constraining, and anticipated the precise nature of the user goal too rigidly. The subjects wanted and expected to be able to perform manipulations themselves. This was demonstrable in goal-tree analysis as a model with too few steps (and too crudely defined steps) reiterates Payne's (1991) emphasis on conversation-style dialogue. The interface should respond to a user action, and in doing so offer affordances for further action. However, a feature which goes beyond the scope of the request implied by a user action, is potentially problematic. This is compounded by a further problem which was exemplified by the subjects' immediate responses. Subjects responded to the unexpected system behaviour by trying manual edits. By the time they had discovered that the system would not respond, they had also lost the option of reversing the placement action, which was available only as an undo for the immediately preceding action.

The problem operating the 'Column' feature was also interesting. The subjects expected to use the standard icon operation on text, even though this did not match their device-independent task-model. This contradicts the proposition that the task-model structure is the dominant influence on user expectations. The subjects' behaviour suggests that recognition of types of operational principles may supersede prior task-knowledge. The subjects' expectations of the columnising sequence [select text-select columns] suggest that they generalised known operations for using iconic features to operate on text.

The nature of task design seems strongly linked different knowledge resources that the user is required to use. With this in mind, it is conceivable that some benchmark evaluation could be performed prior to user testing. The design of individual tasks could be examined to establish the demands that are placed on the user. In the case of Word, the majority of command-based operations involve the selection of an area (a location or a string) and the selection of the relevant feature. Therefore, it can be surmised that the user must know the operation, and be able to match the feature to the

task. This describes operations such as the changing of fonts for sections of text. A task-design that has more than one selection action requires more of the user. For example, the user confronted by the columnising task cannot access this standard [select text--select option] operation. In these circumstances, the user must be aware that an unusual task-step is required, and must be able to search and recognise a suitable feature. The user cannot use known rules, but must use knowledge from the task-domain. Evaluation must decide whether it is likely that the user can do this (the finding in the study was that it is not at all likely).

The Cut and Paste procedure also places demands on the user to comprehend a metaphor, rather than accessing a 'standard' sequence. The user must be aware, from comprehension of the 'cut and paste' metaphor, that two commands are required, and recognise features bearing relevant cue labels. Alternatively, the user must be able to interpret the feedback from the initial action, to know what else is required. The feedback on Word (and numerous packages where this is used) is far from explicit. This suggests that there is a trade-off between naturalness (of the metaphor) and less natural but more economic task-design.

6.11 Summary

In this chapter the diagnostic power of the model of errors has been tested for its coherence extent and limits. The model was unable to provide single specific diagnoses for all incidents. This was partly due to the revelation of multiple problems by some phenotype incidents. However, the need for follow-up analysis of some incidents is suggested. The model categories are shown to be capable of linking phenotypes to one or more diagnosis, leaving open the precise details of optimal redesign. The next two chapters will test the more practical aspects of using the model in user-based evaluation. Chapter 7 describes the development of a practical evaluation method. Chapter 8 will investigate whether the method is comprehensible to non-HCI experts and is helpful in the conception of worthwhile design alterations.

CHAPTER 7--Model-Based Method for Novice Evaluators

7.1. Using Modelling Knowledge for Evaluation

This chapter describes an evaluation method developed from the studies of DM dialogues in Chapters 4 and 5. First we consider the knowledge gained from the studies and model developments that have been made. Then we consider what this implies for the evaluation of DM interface usability for novice users. The intention is to help guide evaluators to the potentially problematic areas of design. Criteria for choosing evaluation approaches on the basis of our findings will then be discussed.

7.2. Model Elements For Evaluation

7.2.1. Overview

As has been described earlier in this and previous work, evaluation of highly interactive DM interfaces involves analysis of the multiple elements that influence single interaction incidents. The aim of usability evaluation involves three distinct objectives. These are:

- the recognition of individual errors (phenotypes)
- the diagnosis of those errors (genotypes)
- making accurate prescriptions for their rectification.

In the case of DM interfaces there is evidence both from the Chapter 4 study and from previous literature (Wright and Monk 1989, Payne 1991) not only of the need for improvements in design, but also of a complex and diverse set of problems facing evaluators and designers. We now summarise elements that condition the approach to evaluation.

7.2.2. Errors in the Context of User Activity

The description of errors and of critical incidents is, in essence, the story of interactive sequences. Diagnosis of a critical incident requires knowledge of the user's overall

model of the system and the current task. The need to see incidents in the context of interaction as a whole is particularly crucial in the case of DM evaluation.

Diagnosis of dialogue breakdowns involves close attention to the point in an interactive cycle where the breakdown occurs. This dynamic task context is necessary for understanding what the requirements of the interface are and why they have failed. In particular, it involves comprehending the influence of the system and interface design on the user's current thinking.

7.2.3. Internal and External Knowledge Synthesis

The environmentally-based nature of DM interaction (see Young et al 1990, Payne 1991) implies the contribution of both knowledge and reasoning strategies acquired prior to current action. The nature of their synthesis with environmental information is therefore the key to understanding and diagnosing incidents. Experimental and theoretical research points to the fact that users proceed by specifying action through a combination of retrieved knowledge and environmental cues. It is commonly accepted that functionally rich DM affords varying paths to user goals. Therefore, the method must provide good coverage of individual features and individual tasks.

The Chapter 4 study suggested that the majority of error types can be diagnosed by considering the history of the interactive dialogue, in particular from the point where an individual task goal is formed. This includes accounts of prior knowledge, prior interaction and errors with distanced effects. The studies also suggest that the first account of an error, and a snap diagnosis may not produce a diagnosis for solution. For example, MacDraw users having trouble with accurate plotting due to the 'grid' restriction merely claimed they found it awkward. Further analysis suggested two possible causes of these errors. One was that the use of the grid as a default was unintuitive, given the users' understanding of the drawing metaphor. The other called into question the labelling of a menu option which afforded the altering of the default.

7.2.4. The User's Knowledge

As the Chapter 4 studies demonstrated, the structure of the user's model is a crucial element of interaction in metaphor-based interfaces. The analysis reported shows the barrier to smooth interaction and user competence that is caused by mismatches between device operations and the representation of the task that is held by the user. In particular, those studies show the potential for serious errors caused by differences between the user's notion of a task's structure and the system's actual structure for an operation (see also Keiras and Polson 1985). Chapter 5 also described how the

content of users' models of tasks affect understanding and expectations. In particular, the understanding of graphical images on icons is closely dependent on semantic binding with the task-space.

Diagnosis of errors requires awareness of how the user comprehends the system. Our studies suggest that user's overall model of the domain, and model of particular tasks is critical to performance. Therefore, an evaluation method must include some way of eliciting the models of task and device that the typical user would be likely to possess.

7.3. Designing an Evaluation Method for Novices

7.3.1. Synopsis

This section discusses the characteristics of a method to bridge the gap between expertise and the novice evaluator. The aim is a 'tool for thought' which teaches the novice how to perform evaluation. The practical problems involved in designing an efficient method require consideration along with the theory and focus of the model.

7.3.2. The Method/Evaluator Interface

The terminology and concepts of evaluation will not be familiar to the novice. Also, the novice will not have gained experience with which to interpret concepts. Therefore the method must provide ways of interpreting designs using model concepts without unnecessary jargon or verbose theoretical baggage.

7.3.3 Task Realism

The evaluator will need a realistic sample of user action on a system in order to provide an accurate analysis. This is because the system should be used in a way which reflects
a real working situation. Therefore, a scenario-based approach is likely to be most suitable. This also gives the evaluator the opportunity to design sequences of task-action that are likely to be typical of the system's use. The scenarios are, ideally, drawn from real working practices.

7.4. Profile of the Novice Evaluator

7.4.1. Introduction

The method is intended to be of use to non-expert evaluators. The requirements of the evaluator are therefore defined by their knowledge of relevant concepts. Another critical issue is the synthesis of the method's instruction and the evaluator's skills and abilities. Two issues are raised here. A method must be appropriate, not only to give evaluators adequate help in evaluation, but also to meet with their approval. The evaluator may not use a method if it is time-consuming, unnecessarily detailed or conceptually difficult, or if their is a large amount of redundant output. All these factors require close attention in a method's design. The following analysis gives a generic description of 'novice evaluators' and their requirements.

7.4.2. Limitations to the Evaluator's Knowledge

The target group will be computer literate and have some working experience of the design process. However, they will not have more than passing acquaintance with Human-Computer Interaction theory or practice (and probably no experience at all). Therefore, in order to have a working knowledge sufficient to proceed, the method must communicate the model in a swift, easily digestible format, either avoiding or explaining technical terminology, and cognitive science concepts.

Another assumption is that the evaluator will be experienced in computing, so computer science terminology will routinely be part of their language. This may make it harder for the evaluator to 'empathise' with the novice user of the evaluated package. For example, the evaluator must distinguish between a system concept and a task concept. Given also that the evaluator may, in a real world system, have contributed to the design, focusing the evaluator on the 'language of the user' must be a key aim.

7.4.3. Evaluator's Working Objectives

The place in the 'market' that the method is designed to occupy is the 'budget' end, where expert evaluators are not available and there is a need to perform evaluation quickly. Therefore a method must be efficient in terms of return on resource/effort consumed. This implies swift tuition in model concepts. Along with this is the problem that time and resources for evaluation are likely to be limited. Therefore, the efficiency of the approach is a key factor.

7.4.4. Skill Enhancement

The evaluators, despite their lack of HCI training or experience, may have an (albeit untrained) eye for usability issues and interface problems, born out of substantial experience of computer use. The evaluator may also have strong opinions on certain aspects of usability. The method should ideally make the evaluator feel that these untrained opinions and insights are being enhanced rather than ignored in favour a mechanical or pedantic procedure.

We can also consider how much skill the naive evaluator may naturally have in, for example, interpreting observed errors. The evaluator may already have a degree of competence in the three evaluation stages (discover, diagnose and repair). For example, a technique which allows observation of real users reporting problems, will make it fairly easy for the evaluator to discover design errors. However, as the studies of MacDraw II suggest, more problems lie in appropriate diagnosis and repair. For example, the user's incorrect selection of a feature may be traceable back to the interpretation of a prior state change, or knowledge of a basic concept. Such critical threads (Carroll et al 1993) may not be directly expressed in verbal protocols or field reports. The users involved will simply give a report of their dissatisfaction or their own (often naive) interpretation of the suspected interface bug. This data requires greater interpretation if it is to be of use. The evaluator requires assistance in the analysis of problems, but should be permitted space for interpretation. The evaluator needs to consider a range of possible diagnoses to specify a solution which addresses the root cause of the problem.

7.5. The Model-Mismatch Analysis Evaluation Method

7.5.1. Introduction

The evaluation method proposed is the Model-Mismatch Analysis method (MMA). The following description discusses the method's approach to the theoretical and practical issues involved in evaluation. The principles behind the method may be placed into modified formats, varying with use requirements and circumstances (see chapter 9). The description that follows describes the elements that are required in a practical evaluation session using the method.

7.5.2. Crucial Elements of the Method

The selected method involves scenario-based co-operative evaluation, using user

subjects performing tasks using the package. The user gives a concurrent protocol commentary on their thoughts and actions. The evaluator observes, intervening at critical points for clarification. The user is then given a retrospective interview at the end of the session. Finally, the evaluator studies the data, and recommends design changes if required.

A scenario-based approach is used to ensure that the user subject will explore the system in a way which reflects its real usage in a working environment. Whiteside et al (1988) express doubts about this approach, preferring undirected exploration. However, evidence from the study in Chapter 4 suggests that a whole-task scenario allows considerable scope for system exploration, whilst ensuring that user subjects are providing ecologically relevant evidence throughout the session.

Another advantage of the approach is that concurrent protocols are easily the most accurate account of users' models of the interface and the current task. This is preferred to eliciting users' models independently, prior to the session by, for example, interviewing them (although the method involves eliciting users' previous task and computing experience). In such an interview the 'user' would give an account of device independent task-knowledge. However, this would give no sense of how the user comprehends and restructures tasks with iconic domain and feature representations. By eliciting the user's model 'on-line', the approach can gauge the whole range of influences on user behaviour (with device-independent domain and task knowledge as integrated elements). Another advantage is that, when errors are made, users provide accounts of their reasoning, and may reveal misinterpretations that have origins beyond the scope of the current action. This allows analysis of underlying causes, or critical threads (see Carroll et al 1993).

Retrospective questioning refers users directly to their experiences of the system. Subjects are asked to give accounts of their understanding of problematic features (allowing for learning from errors) as well as clarifying accounts of their behaviour elicited at the time of the incident. This approach helps users to give accurate reports of their reasoning and behaviour, rather than 'idealised' reflections, an inherent problem with self-reporting (see Stevenson et al 1988).

The method consists of three broad phases, namely the tutoring of the evaluator, preparation for a session, and the procedure for conducting a session. Each element will be discussed in more detail below. The evaluator is provided with a booklet which explains the nature of a DM style interface in general terms, in accordance with established theory (e.g. Shneiderman 1987, Norman 1986). The evaluator is then presented with an explanation of model concepts. This is followed by an explanation

of the procedure for conducting evaluation sessions using the model.

7.5.3. Relationship Between the Final Model and the Initial Model

In Chapter 5 the three-level model of user behaviour was described, which incorporated accounts of knowledge use and error types. This initial model provided a comprehensive account of how knowledge use and errors may vary in character depending on the user's experience (both of current interaction and previous packages and devices) and the type of mental act that the user is attempting to perform. User mental acts are described at different levels of interaction after Rasmussen (1986). For example, rule-based search consists of scanning for familiar locations and features. In contrast, knowledge-based search involves the recruitment of task-domain or general knowledge. Figure 5.5. links these diverse user mental acts to the concept of roles. Roles are tokens representing phases in the cycle of an action, and provide a unified description of mental acts and implicit information needs at all levels of processing.

The Chapter 5 models describe user behaviour and knowledge recruitment at different levels of processing. However, the cycle of action has broad similarities across levels of processing. Roles can be described as abstract tokens representing the sequence of mental acts that compose an action cycle. More specifically, they represent the mental acts that the system should support in display-led interaction. The final model uses the concept of roles to describe the cycle of action. This limits the diagnostic search space for evaluators, and emphasises the concept of the display as an important facet of user cognition. In user-based evaluation, role analysis is the exercise of identifying the point in a dialogue where the user and system models diverge. In turn, roles identify baseline criteria for redesign or remedy. By abstracting the 'phase' of the action cycle where a breakdown occurred, evaluators can establish criteria for the redesign of interface features and rating of possible solutions.

Figure 7.1.a. shows how the concepts described in Chapter 5 relate to the model which is presented to evaluator subjects in the method booklet (shown in Appendix A). The diagram shows the sequence of roles on the left-hand side. Roles are linked to types of mismatch, shown in the next column. These in turn link to mental acts from Figures 5.2.a.b.c. with the associated user processing levels for each mental act described on the far right-hand side. The link between roles and mismatches expresses the divergent nature of user problems associated with each phase of an action. Each mismatch maps to the mental act with which it is associated. The mismatches/mental acts are identified with one or more processing levels, as shown in the far right-hand column of Figure 7.1.a.

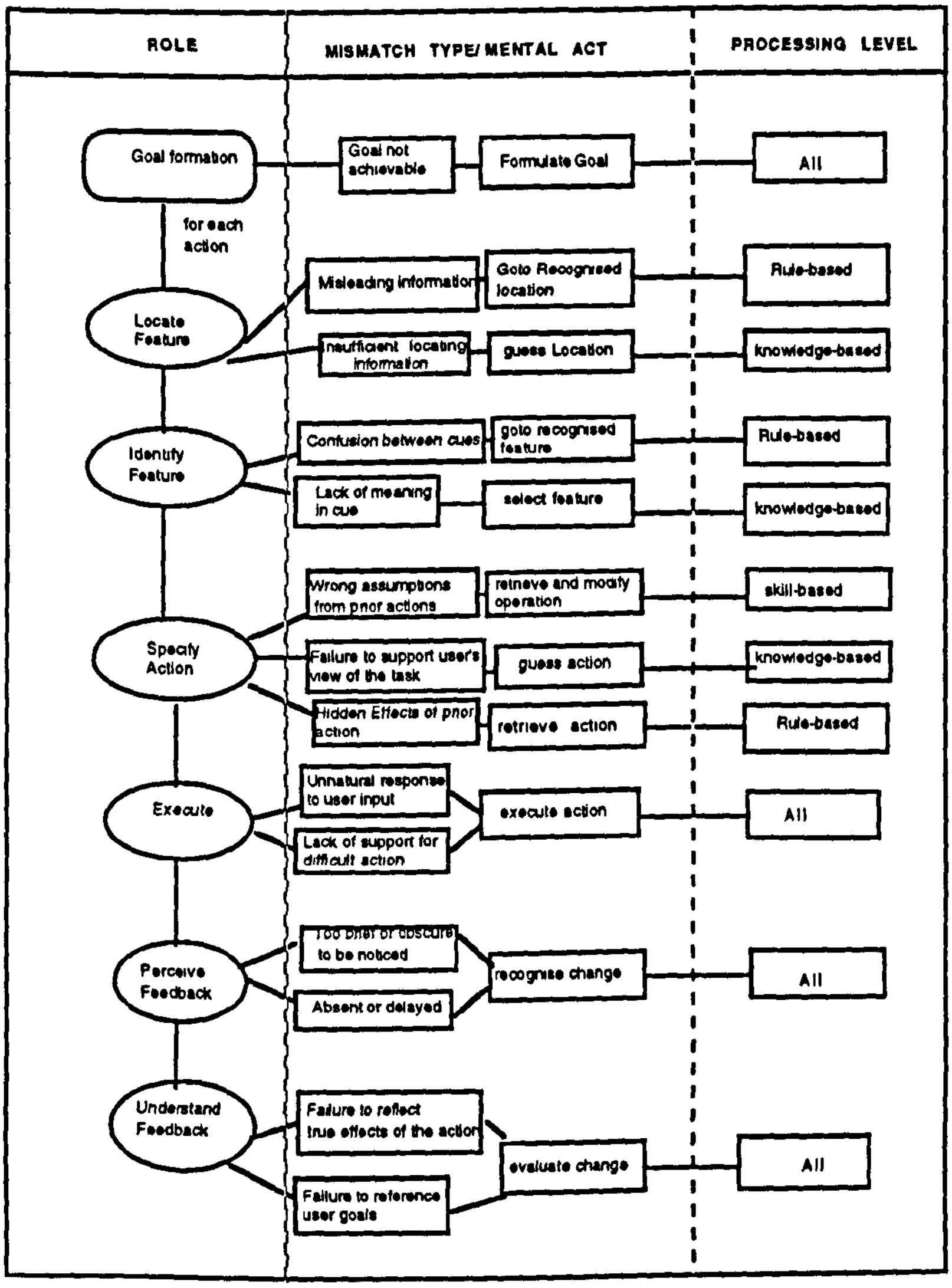


Figure 7.1.a. Links Between Final Model Concepts and Concepts from the Model Described in Figures 5.2.a.b.c.

The alternative mismatch types for each role represent problem types associated with particular processing levels. The 'specify action' role refers to all three levels. This is shown by the three types of associated mismatch which link to mental acts from the respective models. The 'locate feature' and 'identify feature' roles refer only to the rule-based and knowledge-based levels. This reflects the absence of feature search at the skill-based level. The execute and perceive/understand feedback roles cover all processing levels. For example, execution problems may impair interaction at any level. A novice at the 'knowledge-based' level may give up on a feature if it is awkward to use, and the same manipulation may also slow 'skill-based' interaction.

Roles link mismatch types from different processing levels. For example, the 'specify action' role links the hidden effects of a prior action (rule-based) and failure to support the user's view of the task (knowledge-based) as occurring at equivalent phases in the action cycle. Broadly, Norman's (1986) position is that search or simply recognition of environmental cues precedes the specification of operations at all levels. Even at the skill-based level, action is first triggered by recognition of a familiar task-state, although recognition leads to the use of automated procedures. Whilst the models in figures 5.2.a.b.c. implicitly retain this assertion, the three processing-level models do not explicitly emphasise the broad commonality in the sequence across levels. Previous work (e.g. Polson et al 1992) demonstrates the utility of emphasising a common sequence of user mental acts to evaluators. Therefore, the final model uses a common sequence (roles) as a pivotal concept.

The descriptions in the final method use the sequence of roles and mismatch types, along with criteria for identifying examples. Explicit references to the processing level models are not made in the materials presented to study subjects. However, the mismatch types linked to roles correspond one-to-one with mental acts described in either 5.2a.b. or c. and are therefore defined by their associated processing level.

There are two reasons for only making implicit reference to processing levels in the final model and the study materials. The first reason is that evaluators, under any circumstances, will have much to learn when introduced to the approach and the learning burden should be minimised where possible. The theoretical model is too complex to present to evaluators in the initial stages. The sequence of roles truncates the baseline knowledge required by evaluators (i.e. a description of the display-led action sequence). Another related issue is the possibility of mismatch examples that are exceptions to the general descriptions in the initial model. For example, it is possible that 'confusion between cues', which is described as a problem at the rule-

based level, could be caused by a poorly designed feature metaphor prompting the user to incorrectly access general knowledge in feature identification.

The second reason is that detailed analysis of the skill-based level may not be necessary in user-based studies. Most user-based evaluation is conducted with novice users. Given this, the errors found are likely to be either at the knowledge-based or rule-based levels. Therefore, the critical focal point will be at these levels. The model in Figure 5.2.c. would largely be redundant. Given that awareness of the theory of skill-based learning may mean substantial extra learning for novice evaluators, it is worthwhile removing it for their initial exposure to the method. However, they may use the Chapter 5 model for sophisticated analysis as they become more experienced.

7.5.4. Explanation of the Model

In Chapter 5, a model was described in which the central means of analysis was a sequence of dialogue roles that interface must fulfil for competent interaction to proceed. The initial model that is presented to the evaluator is designed to introduce the notion of action as a sequence of mental acts associated with the observable physical acts. The model outlines action at a three-stage level of granularity (specification, performance and evaluation) with nodes describing the roles linked to appropriate places in the cycle. This model is shown in figure 7.1.b. The theory is introduced by describing user mental and physical activities within the cycle of action, and associating them with roles.

Some alterations were made to the presentation and terminology of model concepts. The motivation for the changes was to make key concepts simpler to learn and understand for novices. Mental activities in the cycle correspond one to one with the roles. This is expressed in Figure 7.1.b. by naming roles in a way that gives them a common token with activities. Therefore, the 'locator' role, for example, is referred to as 'locate feature'. Also, the 'operation specifier' role is referred to as 'specify action'. A further terminology change is the replacement of 'role failure' with the term 'mismatch'. This is to express the nature of the technique by which role failures are established (the comparison of user expectations and the system model).

The concept of mismatches is then introduced with full descriptions and examples. Mismatches are described along with their association to dialogue roles. These are augmented with practical examples of mismatches. Each mismatch type is attached to an appropriate activity/role in a flow-diagram which is shown in figure 7.2. The accompanying text describes the nature of associated mismatches. Brief examples of

each type of mismatch are then offered along with advice on how to identify each mismatch from protocol evidence. A full example of the Introductory booklet is provided in Appendix A. Below is a description of the advice on identifying error types along with an explanation in each case.

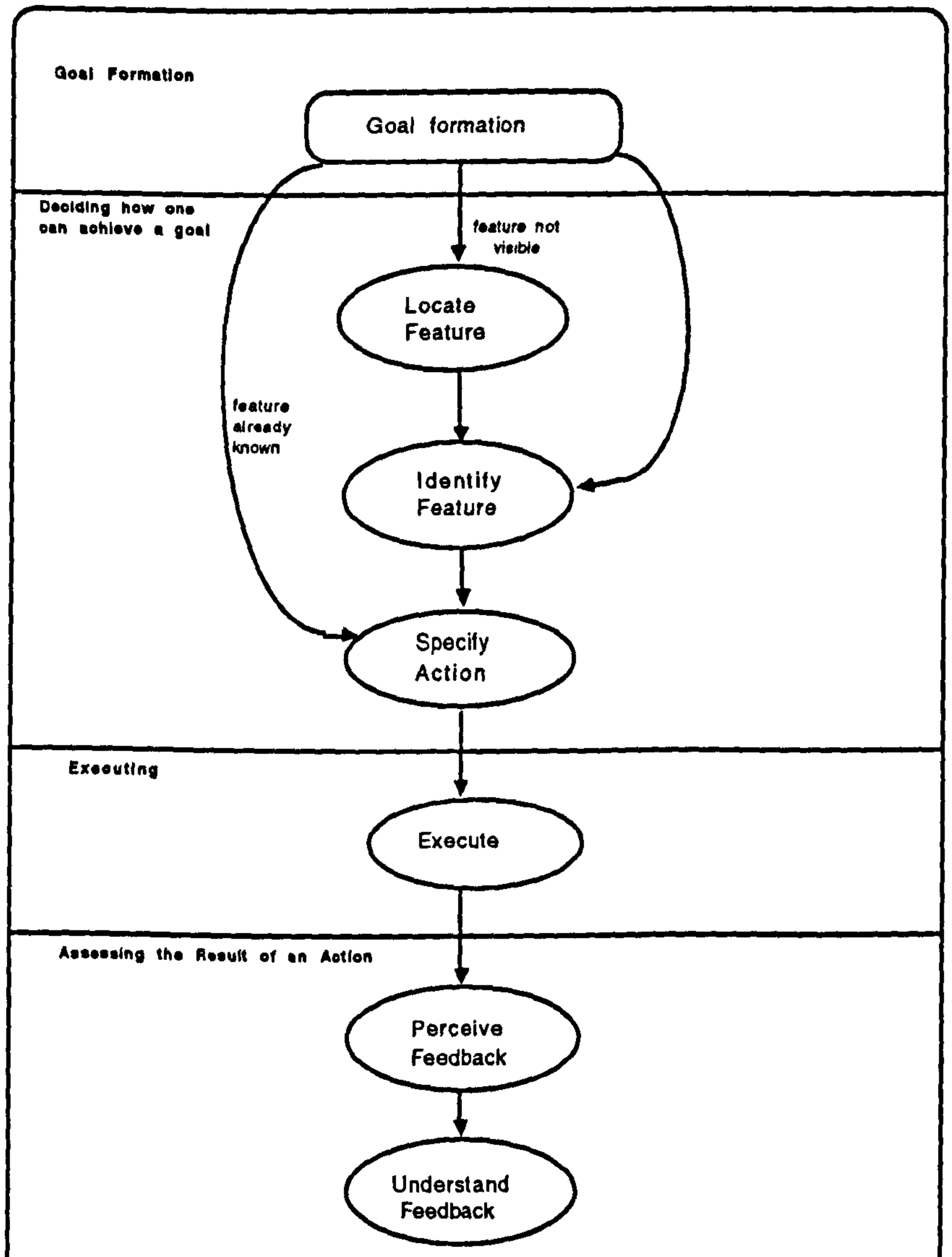


Figure 7.1.b: A Four Stage Model of Action With Dialogue Roles Assigned

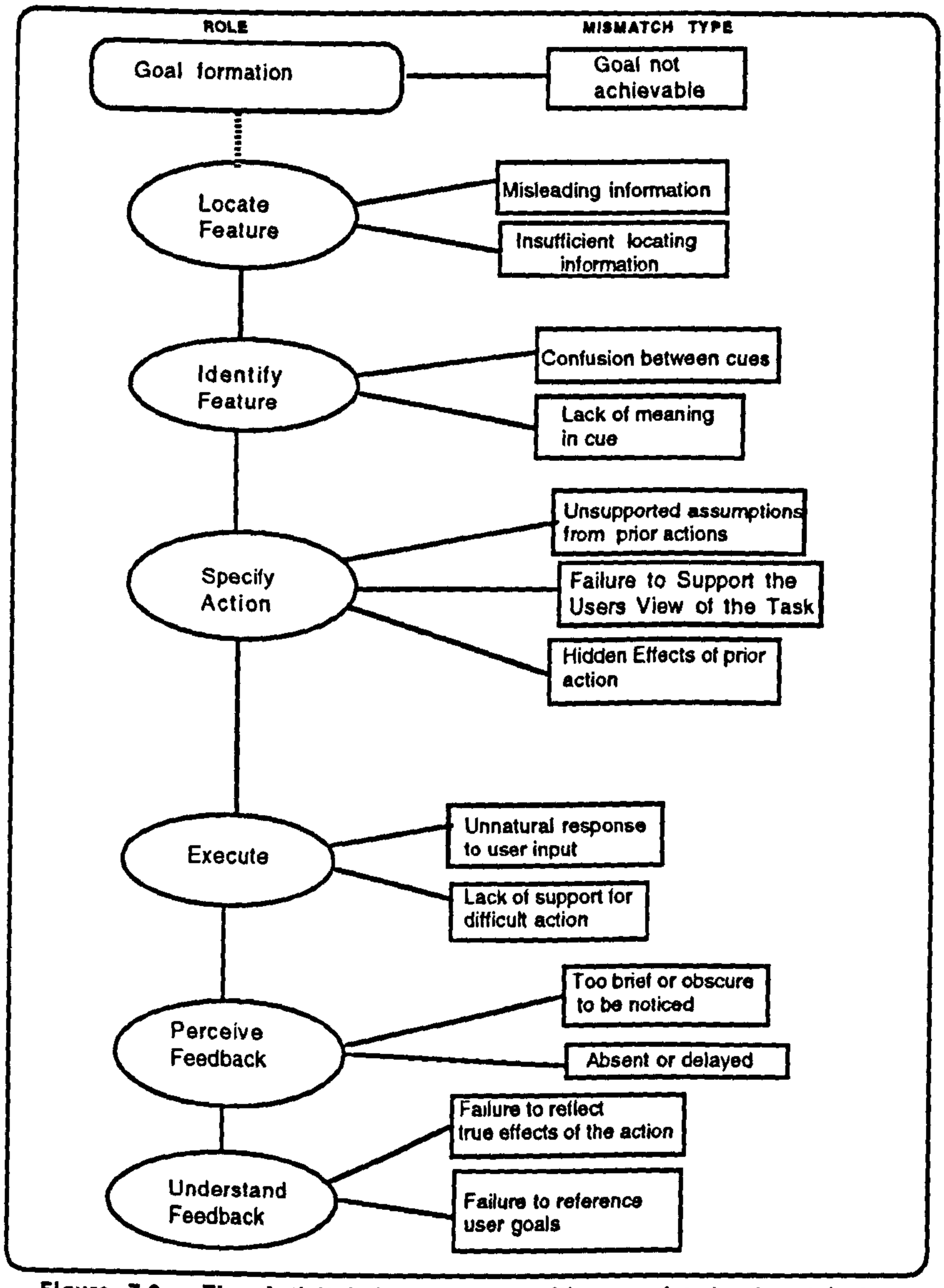


Figure 7.2. The Activity/role sequence with associated mismatch types

7.5.4.1. Goal Formation Mismatches

These may be identified by the user declaring a task-goal that is not, in fact, supported by the interface. This is distinct from a problem that was more prevalent in the Chapter 4 study, namely goals that were possible, but not by the expected means. A user also may express disappointment with the effects of an action, despite correctly identifying and operating the feature, and correctly interpreting the result. These are cases where the feature fails to support the specific goal that the user had in mind, despite being relevant to the task type. For example, a user may wish to use a four-column format for a text document, and find that the system facility for columnising text only supports a three-column format.

7.5.4.2. Locate Feature Mismatches

The main indicators of a Locate Feature problem tend to cause the user to re-specify action or formulate a new goal, without actually performing an intended action. In these examples the user will search in the wrong place for a feature, perhaps naming an area of the interface, declare difficulty in deciding where to search, or express disappointment with failed search. This links both to knowledge-based and rule-based search. The user may not be able to interpret the system metaphor. Also, the user may expect a feature to be in a particular location (e.g. the Show Clipboard feature in the MacDraw Edit menu) and abandon search when the feature is not found.

However, in some cases the user may proceed with an action before the error emerges. The Locate Feature mismatch may lead to an erroneous feature selection. If the user's account of feature choice indicates an ignorance of the 'group identity' for example, of a menu, this is likely to be a Locate feature mismatch. This is illustrated by the example in Chapter 5, of Subject C selecting 'Reduce to Fit' for point size reduction. The feature was in a menu dedicated to general default setting and viewing options. Subject C interpreted the cue without this contextual knowledge.

7.5.4.3. Identify Feature Mismatches

The user may select a feature incorrectly (or verbally misidentifies a feature). In this case the error emerges once the action has been performed (or may be indicated in the user's verbal declaration of expectations prior to action). This is similar to some Locate Feature mismatches. The crucial difference is that the feature representation itself (rather than confusion over group headings or clustering) that is the problem. The distinction is useful because it points to alternative options for redesign.

However, some Identify Feature mismatches are detectable by the user declaring the need for a feature, searching for and scanning the feature, but failing to recognise its

cue. This refers to the problem of cue discriminability, described in Chapter 5.

7.5.4.4. Specify Action Mismatches

This category covers three types of mismatch. These will be described in turn.

• Unsupported assumptions from prior actions: The user may make generalisations from previous actions, making assumptions about the effect of actions and the legality of operations. This could be as a result of the features being cued similarly, located together, or being (to the user) part of a homogeneous group. This refers to the failure of the system to support rule-based specification of operations ('retrieve and modify operations'). Some operations may involve extra operations, or a different ordering of operations, and therefore confound the example-based generalisation.

• Failure to Support the User's View of the Task: This mismatch type refers principally to support for knowledge-based specification. The user may specify, by importing heuristics from the task-space to the device-space, an incorrect or suboptimal sequence of actions. This often involves missing out steps, or applying the wrong operations. This can be explained as a mismatch between the task in the user's model and the task on the device. The user may reveal this problem by stating assumptions drawn from the task-domain or general knowledge. The user may also indicate this problem by describing the system state, or the intended actions, without reference to a key step or concept.

• Hidden effects of prior action: This error tends to effect users performance on features used before. The user may have (unknowingly) altered some aspect of the system state which makes a feature behave differently. As a result a used feature behaves differently, with the user reporting inconsistent behaviour.

7.5.4.5. Execution Mismatches

These mismatches refer specifically to physical manipulations. For example, a user may express frustration when a drag action intended to stretch an object causes it to contract.

Also, there may be actions that, by nature, tax the user's manipulation skills. For example, the user will find drawing a straight line difficult without extra support (a ruler in the case of pen and paper drawing). Dissatisfaction expressed at the performance of a feature may be traced to this lack of support for difficult action.

7.5.4.6. Perceive Feedback Mismatches

These problems may only emerge in the action specification stage of a subsequent action. The user may miss feedback information, and make erroneous assumptions

which affect further actions. As will be discussed below, some of these mismatches could also be classified as Action Specification problems, because they represent a mismatch between the end state of action that the user expects, and the actual state. The automatic default setting problem described in Chapter 4 comes into this category. The errors may only be clear to the evaluator after the user has responded to subsequent action where a hidden state change has taken effect. Therefore the method alerts the evaluator to the connection between 'Perceive Feedback' mismatches and 'Hidden Effects of Prior Action' mismatches in subsequent action.

The other indication of Perceive Feedback mismatches is the user directly complaining that there is a lack of information about the effects of the action.

7.5.4.7. Understand Feedback Mismatches

These are often cases of the user mistakenly believing that there is a problem as a result of an action where, in fact, there is not. In these cases the user will express incorrect assumptions drawn from the feedback, or confusion at the apparent result of an action. Here, like some Perceive Feedback mismatches, the user's incorrect assumption may only cause tangible problems when subsequent action is specified and attempted.

7.5.5. The Category Assignment Problem

A potential difficulty with this approach, and one that is common with most taxonomies, is the possibility of an incident fitting more than one category. One of the potential problems faced by the evaluator is deciding between candidate categories. Even with the advice provided, the evaluator may be left with value judgments to make, before selecting the design element that should be altered. This is particularly true in the case of errors where a task has been attempted without a necessary feature being used. The device inevitably restructures tasks in some cases. It may be argued in such cases that a feature cue is inadequate, or that the structure of the task is not appropriate. Therefore, the method stresses that ultimately the evaluator must make a judgment, choosing between alternative assignments. The provision of a comprehensive set of if-then rules for diagnosis and solutions is beyond the scope of this approach. Therefore the instruction booklet emphasises the importance of the evaluator's intuition and judgment. The principle that is encouraged is that the evaluator should consider which of a range of possible design changes would have the most beneficial effect.

7.5.7. Solutions

The method supplies only minimal advice on the choice of solutions. The user is simply given a brief general summary of the sequence from incident recognition through pinpointing of cause to solution formulation. The evaluator is provided with a table to brief them on the typical protocol evidence types associated with mismatch types (shown in figure 7.4.). The evaluator is encouraged to contemplate possible rectification of errors, rather than given procedural rules necessarily linking genotypes to redesign options. This is favoured for two reasons. The first is that diagnosis and solution formulation are sometimes interleaved. The nature of a mismatch may ultimately be pinpointed when the effectiveness of the potential solution is contemplated. Another consideration is that design changes may . It is a useful exercise for the evaluator to consider such effects when deciding between solutions (or if any change is desirable at all). However, the links described in Figure 5.5 may be used as an aid to decision-making. This chart offers both a diagnosis aid and an indication of which dialogue technique (e.g. visual metaphor, support for abstraction-based generalisation) is implicated by a critical incident.

7.6. Preparing For the Session

7.6.1. Scenario Selection

The favoured scenario consists of a visually displayed task on a paper sheet, showing the start and end state of a task. This is intended to minimise the amount of explicit instructions that the user is given. This approach is favoured to giving the user a sequence of small atomic tasks (e.g. edit an item within a table, move the largest text item). The 'whole task' approach allows the user scope to perform a high-level matching between task and device space, which is something that any new user would be faced with. Thus the user is free to order sub-tasks in a natural way, allowing evaluation of sequences of task-action. This, in turn, is more likely to reveal serious errors which can be traced beyond a single sequence of task-action, such as problems associated with the overall system metaphor or the delayed effects of misleading feedback.

The scenario must be representative of typical tasks that the system would be required to support. This may not be a trivial task with the more specialised packages. Consulting members of the target user group to validate scenario design is advised in such cases.

It is possible that the other main objective of scenario design, testing the range of functionality offered by the system, may conflict with the 'typical tasks' objective. Typical tasks may leave some of the more advanced or specialised functionality untouched. The more specialised functionality may require a scenario that is too demanding of a novice user subject, particularly in a timed experimental session. This could be overcome by giving a second scenario to the user at the end of the main session. Alternatively, the evaluator could 'walk through' more complex task scenarios with the user, asking questions such as 'which feature would you use for this action'.

7.6.2. Preparation of a Training Schedule

The user subject will need to be given basic training in high-level system principles and manipulations. Typical or standard feature operations can be used to demonstrate basic principles. The user is then given time to explore the system (in advance of seeing the scenario). This is both necessary and useful to the evaluation. The necessity is that the user must be acquainted with the basic manipulations to allow concentration on dialogue problems. The guidelines provided, for example, by Apple, are that their products need only a brief demonstration-based tutoring session to set up learning by exploration. This also has the advantage of exposing inconsistency errors such as 'wrong assumptions from prior actions'.

7.6.3. Selecting Representative Users

The user subjects recruited for the sessions should broadly be representative of the target user group. The crucial factors are educational background, familiarity (or not) with the operating platform, general computer experience, experience of the domain and nature of work experience. Ideally, the users should be a sample of the target user group, although there may be practical problems such as availability.

7.6.4. The Session

The user is given a set of instructions as illustrated in Appendix Y. The user is asked to perform the task providing a continuous verbal protocol, similar to the method described by Ericsson and Simon (1983). The evaluator reads the task instructions to the user, along with the requirements for providing a verbal protocol. The protocol may be audiotaped to ensure accuracy. It is also good practice to reassure the user that the package is being tested rather than the user. The evaluator is advised to study the user's behaviour, intervening only when the the user expresses a problem with specifying and performing an action, or with the result of an action. The evaluator

records the concurrent user account of the incident on an 'incident record sheet', an example of which is shown in figure 7.3. Where facilities are available it is useful to make a video recording of the session. A recording of the incident can be played to the user after the session, to help them remember the incident.

The sheet records information which allows the evaluator to infer the user's model of the task. The information may be volunteered by the user naturally in protocol form. The evaluator is advised to intervene and ask the user to clarify their account of their expectations, and the problem as they see it.

Description of Incident							
tries to put arrow on arc line using menu option. claims 'it worked before' having done it for straight lines (can only be done by constructing a line from arc & straight lines, adding an arrow)							
Repeats of same problem	/	/	/				
List any alterations/corrections?							

Figure 7.3. Incident Record Sheet for use by Evaluators

The form also contains sections to help the evaluator gauge the severity of the problem. There is a set of boxes in which the evaluator can record the number of times that the user repeats an error. A frequently repeated error may turn out to be a serious design problem for all classes of user. Also, the evaluator is instructed to record supplementary information about the user's response to the error. So if an error is dealt with easily, there may be less cause for reviewing the design than when a protracted error and repair cycle is observed.

When the session is complete, the evaluator may follow-up by asking the user clarificatory questions about the session, updating the Incident Record Sheet where appropriate. After the user leaves, the evaluator analyses the critical incidents,

specifying the features that could usefully be changed and the nature of those changes.

7.7. From Diagnosis to Practical Solutions

The method is intended for use in industry. Therefore some attention has to be payed to the real-world problems that may be faced. One problem is that there may be insufficient time to make significant changes to a developed system. The second is that equipment such as video and playback facilities may not be available to evaluators.

One of the major issues in evaluation is the role that it plays in the design process. Evaluation of the finished product may come too late for changes to be made. However, the method may work equally effectively on prototypes, leaving scope for major changes to be made where necessary. However, the evaluation may still be useful at a late stage, even though changes may be difficult. The exercise is that of pinpointing areas of the designed where user mental activities are not properly supported. This demarcation of the problem-space can inform the design of help facilities, manuals and user training. It shows where there is a shortfall in system support. This reflects the theme of supporting user information requirements, described by Hartson et al (1990) and Mayo and Hartson (1993). For example, the analysis will expose parts of the system that are difficult to learn using example-based generalisation. These areas could therefore be given special attention in a training programme.

Real-world organisations may find it difficult to acquire video/playback facilities to use in evaluation. This may make analysis of incidents a harder. For example, the evaluator would not be able to check that all error incidents had been recorded. However, the method does support the concurrent collection and subsequent analysis of data. This is the format in which it is tested in Chapter 8, where video playbacks are not available to evaluators.

7.8. Summary of Chapter

This chapter has described a practical, model-based evaluation method. The method employs the model of dialogue roles linked to mismatch types as a theoretical base. This model serves as an aid to diagnosis by helping the evaluator to interpret critical incidents. Practical advice on conducting user-based evaluation sessions is provided, incorporating a concurrent intervention technique, and a strategy for retrospective analysis.

The following chapter tests the effectiveness and efficiency of the method in practical

evaluation sessions. The method is compared to a commercially established method, the Usability Checklist (Ravden and Johnson 1989). Novice evaluator subjects are given the task of using one or approach to produce design solutions.

Mismatch Type		Evidence Type
Goal formation		
Goal not achievable	⟷	User expresses a goal which cannot be achieved by any means
Locate Features		
Misleading information	⟷	User goes to a search location, reasoning about the likelyhood of it containing a desired feature. Desired feature is in a different location.
Insufficient locating information	⟷	User unable to find a desired feature, engaging in random or undirected search
Identify Features		
Confusion between cues	⟷	User tries an irrelevant or suboptimal feature, misidentifying the feature
Lack of meaning in cue	⟷	User expresses a goal implying the need for a particular feature, scans and passes over a needed feature
Specify Action		
Unsupported Assumptions from prior action	⟷	User attempts wrong operation (or operations) declaring expectations based on previous action, or use of another feature
Failure to support the user's view of the task	⟷	User describes expectations as to how a task can be performed which reveals ignorance of a crucial system concept or a necessary component action
Hidden Effects of prior action	⟷	User describes expectations which reveal ignorance of a mode or state change caused by a previous action
Inappropriate Functionality	⟷	User describes expectations which include correct feature identification, but is unable to achieve the current goal using the feature
Execute action		
Unnatural response to user input	⟷	User activates input device. Cursor responds by in a manner counter to expectations (e.g. scroll moving faster than the mouse action)
Lack of support for difficult action	⟷	User finds a manipulation is beyond horizons of physical skill (e.g. drawing straight line freehand). System fails to provide feature to overcome problem
Perceive Feedback		
Too brief or obscure to be noticed Absent or delayed	⟷	User causes a state-change, but expresses or demonstrates ignorance of the change, often by specifying subseqent action without reference to the change
Understand feedback		
Failure to reflect true effects of the action	⟷	User interprete the screen image incorrectly
Failure to reference user goals	⟷	User acknowledges a change has occured, but is unable to confirm that an action is satisfactory

Figure 7.4. The mismatch types linked to typical protocol evidence

Chapter 8-- Comparative Testing of the Model-Based Method for Novice Evaluators

8.1. Introduction

This chapter describes a comparative study in which the method described in Chapter 7 was tested against an established method. This method derived from the model of action is named Model Matching Analysis (MMA). The established method is the Usability Checklist Evaluation Method (Ravden and Johnson 1989). The chapter starts with a description of the Usability Checklist approach. The contrast between this approach and the approach taken in the Chapter 7 method is then discussed. The ten sessions reported in the Chapter 6 study were also used to study the performance of novice evaluators using the MMA method. Ten additional sessions were conducted. These were used to study the performance of novice evaluators using the Usability Checklist method. This chapter begins with a description of its design, along with a description of the experimental sessions, carried out by independent, video-assisted analysis. The two sets of evaluation output are then compared for effectiveness, efficiency and user satisfaction.

8.2. The Usability Checklist Method

8.2.1. Principles

The Usability Checklist (UC) is a highly structured way of recording subjects' experiences of a system by answering a series of questions. The method's authors claim that it is a flexible approach that can be used with or without scenarios, and by a variety of subject types. Among the permutations is evaluation conducted by a supervising evaluator who monitors a novice user. It is the user who completes the checklist for subsequent interpretation by the evaluator. It is the method's utility for this format that was tested, although other aspects of its overall utility and suitability were subject to examination.

The main body of the checklist consists of nine sections based on key principles of usability. These principles are accompanied by lists of between approximately 10 and

15 relevant questions. The subject examines the system, either by performing a task or random exploration, making freeform notes as required. Each question is accompanied by tick-boxes to be filled in by a user subject. For example, the following question appears in the 'Explicitness' section:

'Is it clear why the system is organised and structured as it is?'

This is followed by four tick-boxes [always, most of the time, some of the time, never]. Next to these is a box in which the subject can add comments if appropriate. The user has the option of putting 'don't know or 'not applicable' in the comment box where appropriate. At the end of each section are summary questions, eg:

'Are there any comments (good or bad) you wish to make regarding the above issues?'
and
'Overall, how would you rate the system in terms of (explicitness)?'

which is followed by another row of tick boxes [very satisfactory, moderately satisfactory, neutral, moderately unsatisfactory, very unsatisfactory].

Section 10 of the checklist allows the user to express whether a range of usability problem types have been encountered. This section has a similar format to sections 1-9, except the check-boxes are [no problems, minor problems, major problems]. Section 11 allows the subject to report best and worst aspects, along with other general points about the system. The responses provided by the subject provide the evaluation data output.

8.2.2. The Checklist Method in a Split Role Format

This section describes a session where the roles in a checklist are split into observer and user. The evaluator will have a comprehensive description of the usability principles which form the first nine sections of the checklist. Section 9 was left out, in accordance with the authors' recommendations that irrelevant sections should be removed. The section referred to help facilities, which were not provided by the tested version of the package. The checklist is accompanied by a detailed explanation of the procedure for briefing the user and conducting the session. Each checklist has a corresponding definition supplied. The user can ask for a definition of a question.

The user's role is to provide opinions and reactions in the checklist format to be passed on to the observer. The observer then reads through the checklist interpreting the responses that the user has provided. The observer then assesses the implications for the design. The observer also watches the session, and may merge the findings of the user with his/her own observations.

8.2.3. Contrasts Between the Methods

8.2.3.1. Evaluators Role During the Session

To remove ambiguities, the observer in a Usability Checklist session will henceforth be referred to as 'the evaluator' although Ravden and Johnson (1989) sometimes refer to the actual user as the evaluator. The subject performing the actual task in these sessions will be called 'the user'. The Usability Checklist method does not ask the evaluator to play an interactive part in the test session. The evaluator simply observes the process, and calls a halt to the session at the appointed time. By contrast, the Model-Based method advises the evaluator to use a criteria-based intervention technique to clarify user accounts of critical incidents.

8.2.3.2. The User's Role During the Session

The user in the MMA session has to provide a continuous verbal protocol during the session, and is particularly directed towards expressing expectations of the current action, and reactions to the results of search and action. This may be augmented by clarification questions from the evaluator, as stated above. By contrast, the Usability Checklist user performs without verbal feedback, and is asked to make notes (where appropriate) on his/her experiences as they are performing the task. A pen and paper is provided to facilitate note-taking.

8.2.3.3. Evaluator's Role After the Session

The Usability Checklist evaluator's first duty after the session is to act as a 'talking glossary', explaining terminology and concepts in usability checklist questions when requested by the user. The evaluator does not question the user directly. The evaluator then reviews the checklist for information supporting design recommendations.

The Model-Based evaluator asks retrospective questions about the critical incidents found in the session, and is far more responsible, therefore, for setting the issue agenda (although this should be guided by the critical incidents generated in the

session). The evaluator then checks the account of the incidents against the role/mismatch model provided (figure 7.2) to help clarify the root cause of the problem that the user reported.

8.2.3.4. The User's Role after the Session

The Usability Checklist user has a total of 161 checklist questions to address. The user also has the option of filling in prose sections describing experiences. The MMA method user is not required to give any written accounts of the session, nor complete any written questionnaires. The user is simply interviewed by the evaluator.

8.3. Study Design

The task for the user subjects was as described in Chapter 6, using the scenario described in Figures 6.1.a and b. The ten sessions described in Chapter 6 were each examined by an evaluator subject, testing the Model Matching Analysis(MMA) method. Ten more user subjects were recruited for the Usability Checklist(UC) sessions. The subjects will henceforth be referred to by their group number along with the acronym for the method tested in their session, and their role in the session (user/evaluator). So user MMA5 is the user subject in session 5 testing the MMA method. Evaluator UC3 is the evaluator subject in session 3 testing the Usability Checklist.

8.3.1. Recruitment for the Study

As stated in Chapter 6, the 10 sessions used for interaction and error analysis were also used as sample sessions using the MMA method. Another ten sessions were held using the same scenario and conditions. Twenty students at City University were given the role of the Evaluator. All were students on computer based courses with between 2.5 and 4.5 years computing experience. Ten of the subjects (five using each method) had done a single-term course in Human-Computer Interaction. The other ten had no HCI experience.

The user group were recruited from the School of Social Sciences at City University. The user group all had less than two years experience of using word processing packages. None had ever used any version of Microsoft Word.

8.3.2. Training for Evaluators

Each evaluator was presented with a copy of the method they were to use and a set of instructions for conducting the session. The materials were given to them one day before the session was scheduled in order to standardise the time spent studying them. Appendix A shows The MMA method package. Appendix B shows the prepared instructions for the Usability Checklist package. These were presented along with Sections 2,3, and 4 of the UC method book by Ravden and Johnson (1989).

8.3.3. The Sessions

The evaluators were asked to study the method, and any questions they had were resolved. The evaluators were responsible for briefing the users about the session by reading standard sets of instructions to user subjects at appropriate points. However, training of users, and supervision of the training period was conducted by an independent observer who was present at each of the twenty sessions. This was done to ensure that the training of user subjects was consistent.

The procedure for the sessions was as described in Chapter 6. Both sessions were timed at 30 minutes. There was no time limit on the retrospective analysis with the user, as the time taken to complete this phase was being investigated. The evaluators were asked to analyse the data and make design recommendations before leaving the room. Again, the time taken for the final analysis phase was scrutinised. Both user and evaluator subjects were given retrospective questionnaires eliciting their attitudes to the method and the task.

8.4. Results

The first analysis studied the actual performance of user subjects in the sessions. As the ten MMA sessions are reported in Chapter 6, the account here is relatively brief.

8.4.1. Actual errors made during sessions

The sessions were independently analysed on video to establish the number and nature of the actual errors that were made during the session. This was augmented by retrospective questions to the user subjects, conducted independently of the methods. The errors were classified according to the task that the user was attempting at the time. The scenario contained 6 distinct tasks. Along with these were scanning and repair, included as separate categories.

The twenty sessions produced a total of 186 errors. Of these errors, 99 occurred in the UC sessions and 87 in the MMA sessions. A t-test was conducted on these totals, and found the difference was not significant at P = 0.162. Figure 8.1. shows the totals per task category. The 'columns' and 'bullet' tasks were the most problematic for both groups of subjects.

	MMA Subjects											UC Subjects											
	1	2	3	4	5	6	7	8	9	10	T	1	2	3	4	5	6	7	8	9	10	T	O/T
columns	1	3	2		4	1	2	3	3	3	22		1	4	3	1	3	4	4	2	3	25	47
bullet	2	2	1	3	2	2	2	2	3	2	21		4	1	1	3	3	1	1	1	4	19	40
change	2		2	3		1	1		2		11	1	2	1	1		2	3	2	3	1	16	27
move	2	2	1			2	2		1	3	13	2		1		2	1	1	1	1		9	22
centre	1	5		1			1	1	1	1	11	3	3			1					1	8	19
repair					3			1			4		2		1	3	1	2			1	10	13
enter		1		1		1				1	4	1		1	1	1		1	1	1		7	12
scan				1							1			1	2				2			5	6
total	8	13	6	9	9	7	8	7	10	10	87	7	12	9	9	11	10	12	11	8	10	99	186

Figure 8.1: Errors by Task for MMA and UC Subjects

Ten errors by the UC subjects were failed attempts at repair, compared to only three for MMA subjects. Also, five errors were made during scanning which was not overtly part of a declared task.

The errors by UC subjects were classified using the phenotype categories described in Chapter 6. These are shown in figure 8.2. They show a relatively low incidence of failure to find features. There were no significant differences between the MMA and UC subjects (P = 0.483, Mann-Whitney U test).

	MMA users										UC users										Totals		
Phenotype	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10	MMA	UC	overal
Reject Feature	2	1	0	1	3	1	2	1	1	1	2	4	2	5	2	3	1	2	1	3	13	25	38
Unable to find features	3	3	2	1	2	1	2	1	3	3	1	2	0	0	3	1	0	1	0	2	21	10	31
Unsatisfactory result	3	4	4	6	4	4	4	4	6	6	4	6	6	4	5	6	9	4	7	5	45	56	101
Accidents/ manipulations	0	5	0	1	0	1	0	1	0	0	0	0	1	0	1	0	2	4	0	0	8	8	16
Totals	8	13	6	9	9	7	8	7	10	10	7	12	9	9	11	10	12	11	8	10	87	99	186

Figure 8.2: Error phenotypes for MMA and UC user subjects

8.4.2. Analysis of Individual Errors for UC subjects

The individual errors made in the ten UC sessions were analysed further. Figure 8.3. shows all the errors that were made more than once in the ten sessions. The phenotype classification for each error is listed. The most common error (twelve instances) was failure to select, although only three subjects made the error more than once. The column icon was tried by six subjects, all of whom tried to make the feature work for a selected area. They found that Word ignored the selection, and

turned the entire text into a single, narrow column.

Five incidents involved indent for the 'bullet' feature. This feature is operated by selecting an icon, or from a menu. An indent is automatically created along with the bullet. The users did not expect this, and had difficulty dealing with it. Three subjects suspended the bullet placement task as a result of the feedback, and did not return to it. Five subjects also had difficulty editing manually created columns. The automatic carriage return feature proved troublesome for five subjects who tried to edit text within columns after their initial creation. Five subjects also experienced difficulties having altered an item to bold text and continued typing beyond it. They found unexpectedly that the new text being entered also came out bold.

The New Document icon caused four subjects serious problems. When the new document opened over the task document they believed that the text had been deleted (three of the four subjects admitted to believing this had happened). The error led to considerable delays in the task performance of all four subjects, and some secondary errors. One subject brought up another new document by attempting to reverse the selection. Two subjects searched the menus and selected 'find' facilities.

Three subjects tried to select and move text using Carriage Return (resulting deletion of the selected text). The selection of incorrect features in general did not focus on any particular features, the twenty-five observed errors covering fourteen incorrectly selected features.

Error Description	no. of subs	instances	Classification
Failure to Select Before Action	9	12	Unsatisfactory Result
Column icon	6	6	Unsatisfactory Result
Bullets (auto indent/placement)	5	6	Unsatisfactory Result
Auto-text position/ return	5	5	Unsatisfactory Result
Bold menu	5	5	Unsatisfactory Result
New Document icon	4	4	Reject Feature
Select and move by carriage return	3	3	Reject procedure
Bullets	3	3	Unable to find features
Paragraph menu	3	3	Reject Feature
Screen View icon	3	3	Reject Feature
Use of search for file after selection of 'new document' icon	2	3	Unsatisfactory Result
Centring facilities	2	2	Unable to find features
Column making facilities	2	2	Unable to find features
Double select of areas	2	2	Unsatisfactory Result
Draw Package icon	2	2	Reject Feature
Style menu	2	2	Reject Feature
Symbol menu scroll	2	2	Unsatisfactory Result
Difficulty using text select	2	2	Manipulation difficulties
Accidental double paste	2	2	Manipulation difficulties
Justify icon	1	2	Reject Feature

Figure 8.3: Errors occuring on more than one occassion in the ten UC sessions

8.5. Evaluator Performance

The study compared the performance of the two methods in diagnosing design problems and generating change suggestions.

8.5.1. Solutions offered by Evaluators

The design changes offered by each set of evaluators were analysed along with the stated reasons behind their suggestions. The solutions were independently analysed to classify the type of design change that was suggested. Five categories were used to represent distinct types of solution.

Improve Feature Cueing: This refers to the design of feature and action cues. This could refer to a specific feature, or a general recommendation (e.g.' improve icon designs').

Alter Functionality (features): This includes adding a new feature, removing an existing feature, or altering a procedure for feature use.

Alter Functionality (general): This refers to more general principles of operation, such as how menus are operated, and cursor behaviour.

Alter Visual Layout: This refers to the visual metaphor of the system, including icon groupings, page metaphor, and menu organisation.

Help: Refers to any solution advocating the addition of help facilities, or referring to 'adding help'.

Figure 8.4 shows the number of solutions of each type proposed by MMA and UC evaluators. The number of instances of each solution is shown alongside the number of subjects. Nineteen of the the MMA subjects' solutions proposed that the functionality of a particular feature be altered, compared to four from UC subjects. There were eight suggestions of adding help facilities from MMA subjects, compared to twenty-three from UC subjects. These scores were significant at the 1% level ($x^2 = 17$). There were relatively few proposals to alter general operations, five by MMA subjects and two by UC subjects. The same applied to visual layout, with five suggestions from MMA subjects and four from UC subjects. The largest contrast was for the suggestion of adding help facilities.

Type of solution	Total Solutions		Direct references		probable generic references		Possible references		No discernable reference	
	MMA	UC	MMA	UC	MMA	UC	MMA	UC	MMA	UC
Improve feature cueing	12	9	8	1	17	8	0	9	0	1
Alter functionality (features)	19	4	14	3	5	0	1	2	0	1
Alter functionality (general operations)	5	2	4	1	2	0	1	0	0	1
Alter visual layout	5	4	0	0	14	8	3	10	0	1
Help	8	23	1	4	11	30	6	56	0	0
Unclear	0	3	0	0	0	0	0	7	0	2
total	49	45	27	9	49	46	11	84	0	6

Figure 8.4. Total solutions by each set of evaluators, with references to actual errors made by users subjects during the sessions

8.5.2. Sources of Solution Suggestions

The set of suggestions given by each evaluator were examined for their resemblance to events in the actual sessions. The right-hand columns in Figure 8.4. show the results of this study. If a solution clearly referenced a specific error incident it was classified as 'direct reference'. If more than one incident was referred to, or a probable but non-explicit reference was made to an error, each error is scored under 'Probable/generic reference'. If a solution does not make any reference to an error but could be deemed relevant, it is entered as 'possible reference'. Otherwise, the solution is entered under 'no discernible reference'.

8.5.3. References to Actual Errors by MMA Solutions

Twenty-seven of the solutions offered by MMA subjects referred directly to independently observed errors (see Figure 8.4). Of the twelve suggestions for feature cueing improvement, 67% referred directly to a single incident. Of the nineteen solutions in the 'alter functionality for a feature' category, 74% referred directly to single incidents. Four out of five solutions suggesting altering general functionality referred to a single incident.

Seventeen incidents were probably or generically referred to by 'improve feature cueing' suggestions. Secondary effects of failure to find the correct feature, or selection of an incorrect feature, contributed to this total. The other substantial scores here were fourteen generic references (in five suggestions) for the general 'alter spatial layout' category, and eight references from two 'help' suggestions. The spatial layout total consisted mainly of search problems, referred to in general terms by the evaluator. There were ten instances of errors with a 'possible' link to a suggestion. Six of these were in the category 'help'.

8.5.4. References to Actual Observed Errors by UC Solutions

A relatively small number of UC evaluators' suggestions (20%) referred to incidents directly. One of these directly cited an entry in Section 11 of the checklist, which asks for textual replies to questions such as 'what was the worst aspect of the system for the user'. Probable and generic references were also sparse, apart from references by 'help' suggestions which accounted for thirty out of forty-six references (65%). Entries in the help section tended to be, for example, cases where solutions referred to 'numerous problems finding menu options'.

There were eighty-four possible references made. This compared to eleven by MMA subjects. Fifty-six of these were possibly referred to by 'help' suggestions (67%). A number of the 'help' suggestions made reference to the menu or icon bar, thus rendering any error involving either as potentially relevant (assuming there was no further reference or focus). Of the others, nine were in the 'improve feature cueing' category, and ten in the category 'alter visual layout'. The inter-method difference in number of direct, probable and possible references was significant at the 1% level ($x^2 = 45.72$).

Six of the solutions offered by UC subjects made no discernible reference to any of the independently observed errors. Of these, three suggestions made clear references to comments or suggestions made in the user subject's checklist answers. The checklist invites users to comment on any aspect of the interface covered in the questions. Also, question 11.4 explicitly asks if there were any 'irksome' aspects of the system, that did not directly cause problems for the user.

8.5.5. Discussion

The high incidence of MMA solutions directly referencing an incident (55%) was not unexpected as the method specifically encourages analysis of user difficulties, and the redesign of features which caused problems. This is borne out by the fact that 67% of 'Improve feature cueing' suggestions and 74% of 'alter feature functionality' suggestions referred directly to incidents. The five 'alter visual layout' suggestions were more general references, reflecting the high incidence of failure to find features by MMA user subjects. The 17 'probable/generic' references by the 'improve feature cueing' suggestions seems also to reflect the number of secondary errors made as a result of failure to find features.

The UC totals show that 51% of solutions offered by UC evaluators were in the 'help' category. These also were the main contributors to the large total of 'possible references' to errors. This reflects the coverage of the help solutions, typical of which was evaluator UC8 with 'offer extra explanation of icons on request'. Several similar suggestions of adding general help for both the icon and menu bar contributed to the total. Also, very few suggestions (nine) referred to single incidents. Again, this shows a tendency by UC evaluators to make general rather than specific design suggestions.

A possible reason behind the UC subjects preference for 'help' is the lack of checklist reference to individual task features. The evaluators may not have had sufficient data to characterise particular design problems accurately. There is also the possibility that the boundary between user and system error may blurred for UC evaluators. The MMA method provides criteria for identifying a design problem from a user error. If a user error is identified as a problem with a feature, the design of that feature is part of the 'solution space' in the mind of the evaluator. This echoes Booth and Grey (1991) who discuss evaluation methods' role in enabling and directing creative thinking. The results suggest that the UC is less successful in diagnosing user errors as problems with the design.

8.6. Analysis of Reference to Actual Errors in Method Analysis

The observed error phenotypes were compared to the retrospective analysis documents for each method. This was conducted to test the extent and accuracy of coverage. The MMA errors were compared to the incident records that were taken and verified post hoc. The UC solutions were compared to subjects' checklist entries. The scores for direct references (MMA 47/12 UC), probable references (MMA 29/75 UC) and possible references (MMA 7/365 UC) were significant at the 1% level (X^2 = 5504).

8.6.1. References to Observed errors in MMA analysis

The MMA method made a total of forty-seven direct references out of eighty-seven errors (51%). Twenty-six of the remaining errors were in the 'probable/generic' category (30%). Of these, nine were made in the incorrect procedure section. Ten errors were not referred to at all (9%).

Phenotype	Instances of errors		Direct reference		Probable/ generic reference		Possible reference		Errors without reference	
	MMA	UC	MMA	UC	MMA	UC	MMA	UC	MMA	UC
Unable to find features	21	10	13	3	6	18	1	61	1	2
Reject feature	13	25	5	3	5	32	1	112	2	3
Manipulation difficulties	8	8	2	0	4	2	2	10	0	3
Unsatisfactory result	45	56	27	6	16	23	3	182	7	13
Total	87	99	47	12	29	75	7	365	10	21

Figure 8.5: The number of independently observed errors, with the number of references made in the analysis process by MMA and UC subjects

8.6.2. References to Errors in Usability Checklists

The scores for usability checklists could be defined by the type of entry that was made. The scores for 'direct references' were entirely accounted for by textual responses either in Section 11 (which asks for textual descriptions) or by checklist

answers augmented by comments. These also contributed substantially to the section 'probable/generic reference. Other scores in this section were found by linking 'never' entries for certain checklist questions to particular errors. Clusters of related questions given 'never' or 'sometimes' answers by a subject were also marked in this section. Any response other than 'always' to a relevant checklist question counted as a 'possible' reference (assuming it had not contributed to the first two totals).

Twelve of the ninety-nine errors were referred to directly. However, seventy-five of error references were in the 'probable-generic reference' category. The majority of these were accounted for by textual responses, although some were relatively clear references in checklist ticks. For example, user subject UC5 gave a series of 'some of the time' and 'never' responses to questions in the 'Informative Feedback' section. This was referenced in one of evaluator subject UC5's suggestions, although it was not clearly linked to a particular error.

The checklist yielded a total of three hundred and sixty-five possible references to errors. Of these, certain questions were particularly prominent. For example, question 3.14 in the 'Compatibility' section reads 'does the sequence of activities required to complete a task follow what the user would expect?'. Any entry in the 'most of the time', 'some of the time' or 'never' categories would link to any problems in finding features or working out operations that have been independently observed in the session.

8.6.3. Discussion

The MMA scores for 'errors without reference' and 'possible reference' were higher than anticipated. The MMA method recommends that all observed error incidents are recorded, even if the evaluator eventually decides that a change suggestion is not required. One cause seems to be the 'lifespan' of an incident. The two cases of MMA user subjects having problems with the typeprint defaults in the 'style' menu were not recorded. These incidents were rectified within a few seconds. The evaluators' may have assumed that this meant they were trivial errors. Also, the speed with which such incident occurred meant that it would have been relatively difficult for the evaluator to spot and record.

The relatively low incidence of direct references to errors by UC subjects seems to reflect the checklist's greater emphasis on principles and usability concepts, rather than

detailed retrospective analysis of the sessions. The number of possible references in the ten sessions is remarkably high (365). This figure reflects the number of principles and concepts in the checklist that could be deemed relevant to particular errors. This suggests that it is somewhat hard for evaluators to interpret checklist entries that have no accompanying comments. Each checklist entry in the 'never', 'most of the time' or 'some of the time' checklist boxes may be relevant to an incident. For example, Checklist question 2.10 asks 'is the method of selecting options consistent throughout the system?'. A single inconsistent procedure would cause a user to consider making an entry under 'most of the time'.

The checklist seems to have a problem of vagueness. The large total for possible references to errors represents a considerable multi-classification problem. This may be linked to the issues raised in Chapters 2 and 5 about the focus of usability principles. The distribution of possible user references in the checklist suggests a difficulty in expressing issues through reference to usability principles from which the checklist questions are compiled. Also, the repeated possible references may have had the effect of blurring the focus of the evaluator, making important issues hard to recognise.

8.7. Qualitative Analysis of Solutions

The quality of the two sets of solution suggestions was put under further scrutiny. The objective of both methods is assumed, in the current context, to be the generation of design improvements. Therefore, a point of interest is the explicitness of the design suggestions generated by each method. The following section describes a quality analysis based on the explicitness of the solutions.

8.7.1. Investigation of Solution Explicitness

The solutions offered by the two sets of evaluator subjects were examined by an independent judge for their explicitness in addressing issues and recommending changes. Figures 8.6A and 8.6B show the results for each evaluator subject for MMA and UC respectively. The analysis used five categories. These were:

• Solution Specified -- This refers to suggestions which give a clear explanation of how the design should be altered. An example is the solution provided by Evaluator MMA5, 'remove the New Document icon'.

• Guideline -- This describes cases where a suggested solution stops short of giving explicit advice. An example is subject MMA5 suggesting making 'the icon for bullets clearer'. Another example is provided by evaluator UC3, suggesting making 'the table option more obviously for columns'. In both cases the nature of the needed change is made explicit, but the designer is not given explicit advice on what a new design should be.

• Issue Flagged -- This category describes suggestions which are clear about a need, but without a clear suggestion of the nature of any change. Evaluator UC3 provides an example, 'the insertion of bullet points could be made easier'.

• Vague -- Some solutions failed to clearly define the issue being dealt with. An example of this is provided by evaluator MMA10, 'high level tools should be more visible or clearly accessible'. The subject failed to make clear which tools he had in mind, or how they could be altered.

• Wrong -- Refers to any solution which embodies incorrect assumptions about the current design.

	Subjects with HCI knowledge (E)					Novice Subjects (N)							
Solution class	Subject 1	2	3	4	5	6	7	8	9	10	E	N	total
Solution Specified	1	1	4	1	2	3	2	1	4		9	10	19
Guideline	2	1	2	1	2	1	3	3	1	1	8	9	17
Issue flagged	1	3				1		1	1	2	4	5	9
Vague		1		1						2	2	2	4
Wrong											0	0	0
Total	4	6	6	3	4	5	5	5	6	5	23	26	49

Figure 8.6A: Solution accuracy classifications for MMA subjects

	Subjects with HCI knowledge (E)					Novice Subjects (N)							
solution class	Subject 1	2	3	4	5	6	7	8	9	10	E	N	total
Solution Specified		2	1		1	1					4	1	5
Guideline	2		2	1		1	5	2		1	5	9	14
Issue flagged	1		2	1	5			1	4	1	9	6	15
Vague		1		2				1	2	4	3	7	10
Wrong		1									1	0	1
Total	3	4	5	4	6	2	5	4	6	6	22	23	45

Figure 8.6B: Solution accuracy classifications for UC evaluator subjects

There was no significant difference in the number of solutions proposed between experienced and novice subjects (P = 0.60, ANOVA) or between MMA and UC subjects (P = 0.48, ANOVA).

8.7.2. Solution Quality Ratings for MMA Evaluators

The MMA subjects all produced at least one specified solution, with the exception of evaluator MMA10. All subjects produced at least one guideline. The group without any HCI experience (subjects 6-10) managed ten of the nineteen specifications. Of the nineteen specified solutions, fifteen were in the 'alter functionality (features) category'. Three were in 'alter functionality (general)', and one in the 'alter feature cue' category. Eight 'improve feature cueing' suggestions were classed as 'guidelines'.

8.7.3. Solution Quality Ratings for UC Evaluators

The UC evaluators produced only five specific solutions. Fourteen guidelines were offered and a further fifteen suggestions flagged issues. All but one of the specified solutions were offered by the HCI experienced group, the exception being being evaluator UC6 who offered 'include an undo icon'. The inexperienced group produced a higher overall total in the 'guideline' category than the experienced subjects (9/5). This is largely accounted for by evaluator UC7 who produced 5 guidelines (four 'help' one 'visual layout'). The number of 'vague' suggestions was higher for inexperienced subjects, with evaluator UC10 the major contributor.

8.8. Further Analysis of MMA

8.8.1. Coverage and Redundancy

A further analysis of the MMA evaluators' coverage was conducted. The number of solutions offered was compared to the number of entries on the 'Incident Record Sheets' used for concurrent and retrospective analysis of incidents. The total incidents analysed for each subject are shown in figure 8.7 along with the totals for independently observed 'actual' errors. The table includes an incident in MMA session 6 which makes two separate references to an incident recorded once in the independent survey. A total of seventy-one incidents were analysed in the ten sessions. None of the subjects made a solution suggestion for all incidents. Subject seven made a joint reference to two incidents in a solution suggestion. There was no significant difference between experienced and novices in the number of incidents analysed ($P = 0.84$, ANOVA), in number of solutions offered ($P = 0.37$, ANOVA) or in the number of actual errors ($P = 0.66$, ANOVA).

	Subs with HCI experience (E)					HCI Novices (N)							
	1	2	3	4	5	6	7	8	9	10	E	N	Total
Incidents Analysed	8	10	6	5	7	8	6	6	7	8	36	35	71
Solutions Offered	4	6	6	3	4	5	5	5	6	5	23	26	49
Actual Errors	8	13	6	9	9	7	8	7	10	10	45	42	87

Figure 8.7: Number of incidents analysed compared to the number of solutions offered and actual observed errors for MMA sessions

8.8.2. Comprehension of Model Concepts

The subjects were asked to add reasons for their solutions. In particular, they were asked to cite relevant role mismatches. Their responses are shown in figure 8.8 below. Two out of ten evaluators failed to offer any citations. Those that were offered were analysed for their accuracy in characterising the problems that the solution addressed. From forty citations, thirty-three were selected as accurate descriptions of cited design problems. There were no significant differences between experienced and novice subjects in the number of solutions offered (P = 0.37, ANOVA) the number of role citations (P = 1.0, ANOVA) or in the number of accurate citations (P = 0.69, ANOVA). Five of the seven inaccuracies were offered by evaluator MMA2. He classified the problems relating to all his six suggestions as 'lack of support for difficult action'. This is a sub-category of the 'execution support' role. Five of his six solutions referred to problems either finding features or working out how to use them. These should have been classified amongst the locator, feature identifier or operation specifier roles.

	Subjects with HCI knowledge (E)					Novice Subjects (N)							
	1	2	3	4	5	6	7	8	9	10	E	N	Total
Solutions Offered	4	6	6	3	4	5	5	5	6	5	23	26	49
Role citations	4	6	6	0	4	4	5	5	6	0	20	20	40
Accurate citations	4	1	6	0	4	4	5	5	4	0	15	18	33

Figure 8.8: Role citations linked to solution suggestions for MMA subjects

8.8.3. Discussion

The incidence of subjects recording incidents but not providing solution may be explained in two ways. First, the subjects are asked to record user errors, and provide solutions to system errors. For example, evaluator MMA6 wrote 'not a system error' next to three incidents. Subject MMA2 wrote 'one gets used to it' against two errors. Also, whilst a feature may cause some errors, it may still be the best design option available. This may be illustrated with reference to the example of cursor placement. The cursor can be moved with the mouse by placing a marker and selecting. User MMA4 mistook the markers position for the actual cursor position, and typed in the wrong place. Evaluator MMA4 consequently recommended that 'the cursor should always move with the mouse'. Given that the pointer is often moved to the scroll-bar or menus, there must be some doubt that such a change would be an improvement.

The high incidence of correct citations suggests that most evaluators' comprehension of the concept of roles and mismatches was satisfactory (although evaluators MMA4 and MMA10 were unable to provide citations). The majority of evaluators were able to accurately reference the appropriate role.

8.9. Further Analysis of the Usability Checklist

The checklist was examined further to assess its suitability for evaluation. Two major themes were examined. One was the suitability of the checklist for testing design features. The other was the suitability of the checklist itself for evaluation using a sample of user subjects.

8.9.1. Checklist Section Entries

The table in figure 8.9. shows the total number of checklist answers in each of the eight 'criteria-based' checklist sections. A total of 1154 checklist boxes were filled-in by the ten subjects. Of these, 119 entries (16%) were in the affirmative section 'always'. This implied that they were completely satisfied with the system's performance for the issue raised by that question. However, the majority of entries (51%) were in the Most of the Time' or 'Some of the Time' boxes. This presented a problem of focus to the evaluators. An average of 58 answers per subject were in these sections, and not accompanied by comments. Each of these answers required attention, given that the answers implied that the system was less than perfect for the cited aspect. A total of 12% of checklist entries were in the 'not applicable' column. Along with this were 124 cases (11%) of user subjects conceding that they did not

know how to reply. There were 83 instances of subjects requiring explanations of questions during the sessions.

Total Answers for Checklist Categories

section	Always	Most of the Time	Some of the Time	Never	Not applicable	Don't know	Total
1	33	48	36	13	19	11	160
2	62	31	13	1	14	8	129
3	22	50	52	5	17	14	160
4	15	52	53	34	21	5	180
5	4	43	39	23	7	12	128
6	8	33	37	8	12	12	110
7	23	30	16	15	29	34	147
8	23	32	22	19	16	28	140
total	190	319	268	118	135	124	1154
%	16	28	23	10	12	11	

Figure 8.9. Total answers to UC sections 1-8

Further analysis was conducted to establish the degree of consensus between subjects

on the checklist questions that they answered. Figure 8.10. shows the results of a sensitivity analysis of answers given to sections 1-8 of the checklist by UC user subjects. The figure records all incidences of a box being ticked by three or more subjects for a particular question. The analysis showed that none of the questions were given the same response by all ten subjects. Majority consensus was rare, with a typical distribution of three or less per tick-box.

One of the more surprising results was the lack of consensus on entries in the 'not applicable' boxes. This may partly be explained by the varying experiences of users in interpreting their respective sessions. However, a problem with some seems to be varying interpretation and comprehension of questions. For example, two subjects answered 'not applicable' to question 2.3 which asks 'are icons, symbols, graphical representations and other pictorial information used consistently throughout the system?'. The subjects ticked the 'not applicable' box for question 5.11, which asks 'where a metaphor is used (e.g. the desk-top metaphor in office applications), is this made explicit?'. User UC1 added the comment 'I did not notice a metaphor'. Five hundred and eighty-seven (51%) of the entries were in the 'Most of the time' or 'some of the time' boxes. Only six of these entries were augmented with comments. This made the interpretive task of evaluators harder as, particularly in the 'most of the time' examples, the entries may well have referred to several features or incidents.

Section	Always	Most of the time	Some of the time	Never	Not applicable	Don't know	Total
10	0	0	0	0	0	0	0
9	1	0	0	0	1	0	2
8	2	3	0	0	0	0	5
7	1	1	1	0	2	0	5
6	4	5	6	1	0	1	17
5	5	12	9	1	1	3	31
4	6	12	12	2	3	5	38
3	8	32	21	4	10	8	83
Total	27	65	47	8	17	17	181

Figure 8.10: Sensitivity analysis, showing the number of occassions that three or more subjects entered the same answer for checklist sections 1-8

8.9.2. Discussion

The major problem for the UC method seems to be a poor focus. The checklist appears to have two major design problems. One is that numerous questions give scope for a number of possible references, but do not constrain the subject to make specific reference to problems. The sensitivity analysis in figure 8.10. suggests that the analysis may not constrain the subjects sufficiently. Another problem seems to be comprehension of questions. The problem of not understanding questions is dealt with in the split-role format by the evaluator providing definitions. However, there

were also cases of subjects making highly questionable assertions about the relevance (or otherwise) of some questions. The two cited examples of subjects declaring a question 'not applicable' demonstrate this problem. The subjects clearly felt that they understood what the question was referring to, but their responses suggested that they did not. These were clear cases, but there is potential for more hidden cases. For example, a user may be asked a question, think the concept is relevant (but misinterpret it) and make an entry in a tick-box. The evaluator would be unaware that the question had been misunderstood unless an entry in the accompanying comment box suggested so.

8.10. Time taken by the Methods

Along with investigations of methods' effectiveness, the time taken to perform evaluation sessions was also measured. The results are shown in figure 8.11 below.

	MMA Subjects										UC Subjects										Totals	
	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10	MMA	UC
Analysis with user	7	5	8	11	6	7	8	2	3	5	41	40	55	44	44	42	43	53	40	46	62	448
Analysis post-user	14	23	30	17	19	29	25	28	20	16	19	35	22	48	34	7	16	29	11	20	221	241
Total	21	28	38	28	25	36	33	30	23	21	60	75	77	92	78	49	59	82	51	66	283	689

Figure 8.11: Time taken for user collaborative, and post-user analysis

There was no significant difference in the time taken for post-user analysis ($P = 0.48$, ANOVA). However, there was a significant difference beween MMA and UC subjects in time taken for analysis with users ($P < 0.001$, ANOVA). The MMA retrospective analysis averaged 6.2. minutes. This was shorter than anticipated. Only evaluator MMA3 asked the user to retrace task-steps for all recorded incidents (by pointing to screen areas). The most popular approach was simply to ask for clarification of what was on the sheet. It is possible that the relatively low incidence of specified cue alterations by MMA evaluators is related to an underutilisation of this phase. Most user subjects' evaluators failed to consult them about possible alternative cue expressions.

The UC users took an average of 45 minutes to complete the checklist. Several

complained about the size of the checklist, seven seven doing so before starting. Those that complained admitted afterwards that they had initially overestimated the likely completion time, while two suggested that the sheer volume of the questionnaire had probably diminished their concentration.

The overall inter-method difference for the post-user analysis was not significant. The individual differences within each subject group were, however, sizable. Evaluator MMA3 took over twice as long as evaluator MMA1 (producing two more solutions). In the UC sessions, there was a difference of twenty-eight minutes between evaluator UC2 (who produced four solutions) and subject UC2 (who produced two).

8.11. Satisfaction Ratings by subjects

Questionnaires were presented to evaluator and user subjects from both groups. They were asked about the task scenario and package, as well as relevant aspects of the method that they used.

8.11.1. Evaluator Questionnaire

The evaluators were asked seven questions referring to aspects of the methods that they used. Their responses are shown in figures 8.12A and 8.12B. They were asked to state a value between one and seven for each question, with seven as the most positive rating and one the most negative rating. The subjects were also invited to add comments to their ratings if they wished. The average rating was calculated for each method. Overall, there was no significant difference between either experienced or novice subjects ($P = 0.63$, ANOVA) or between MMA and UC subjects ($P = 0.16$, ANOVA).

The MMA method scored 4.9 for ease of use and the UC method 4.6. Four of the MMA and three of the UC evaluators referred to their method as well explained and easy to comprehend. Only subject UC2 complained that the terminology of the checklist was 'verbose'. The same subject also expressed difficulty in interpreting checklist results. Whilst the contrast between average ratings for experienced and novice subjects was not significant for MMA subjects (0.2) the UC novices average rating was 1.2 higher than the experienced subjects.

Both methods scored 4.9. for helpfulness. Two MMA subjects (MMA2 and MMA8)

referred to the methods role in 'focusing' their thinking. Subject UC6 made similar reference to the UC method as 'giving a good picture of what was important'. The split in rating level between experienced and non-experienced evaluators was only 0.2 for UC subjects, but 1.0 for MMA subjects.

The next question 'did the method intrude on your own way of conducting the evaluation?' produced an average of 5.3 for MMA subjects and 4.8 for UC subjects. Subject UC10 expressed frustration that he could not interact more with the subject during the session.

Evaluator's Review: Aspects of thMethod Used

	E1	E2	E3	E4	E5	N6	N7	N8	N9	N10	E	N	OV.
How easy to use did you find the method?	4	7	4	5	5	6	5	5	3	5	5.0	4.8	4.9
Did you find the method helpful?	5	7	5	5	5	6	5	5	2	4	5.4	4.4	4.9
Did the method intrude on your own way of conducting the evaluation?	4	7	3	5	7	6	7	6	5	3	5.2	5.4	5.3
Did you feel that the procedures and instructions in the method were clear?	5	5	4	7	6	5	4	5	5	3	5.4	4.4	4.9
How easy was it to comprehend the terms and concepts in the method?	5	7	7	6	6	6	5	5	4	4	6.2	4.8	5.5
Were you generally satisfied with the method?	5	7	6	6	5	6	5	5	5	3	5.8	4.8	5.3
Did you find generally that the materials given to you were readable?	5	6	6	7	5	4	5	6	5	3	5.8	4.6	5.2

Figure 8.12a: Satisfaction Ratings by MMA Evaluator Subjects

Evaluator's Review: Aspects of the Method Used

	E1	E2	E3	E4	E5	N6	N7	N8	N9	N10	E	N	OV.
How easy to use did you find the method?	3	1	5	6	5	6	3	5	6	6	4.0	5.2	4.6
Did you find the method helpful?	4	5	5	5	5	7	4	5	4	5	4.8	5.0	4.9
Did the method intrude on your own way of conducting the evaluation?	5	4	3	4	6	6	5	6	5	3	4.6	5.4	5.0
Did you feel that the procedures and instructions in the method were clear?	3	4	3	5	5	6	2	6	3	4	4.0	4.2	4.1
How easy was it to comprehend the terms and concepts in the method?	3	4	3	6	6	6	2	3	4	5	4.4	4.0	4.2
Were you generally satisfied with the method?	4	4	5	6	5	6	4	4	4	4	4.8	4.6	4.7
Did you find generally that the materials given to you were readable?	4	5	5	6	6	5	3	6	2	3	5.2	3.8	4.5

Figure 8.12b: Satisfaction Ratings by UC Evaluator Subjects

The MMA method scored 4.9 on average, against 4.1 for the UC method, for clarity of procedures and instructions. There was a 1.0 contrast between experienced and novice MMA subjects (4.2./5.2.). However, none of the inexperienced subjects offered specific criticism on this point. The UC novice subjects rated the method 0.2 higher on average than the experienced subjects (4.2/4.0).

The MMA method scored 5.5 on average, against 4.2 for the UC method, for the question 'How easy was it to understand the terms and concepts in the method'. Again, there was a large difference in the rating between experienced and novice subjects for the MMA method, the average being 6.2 for experienced subjects and 4.8 for novice subjects. Subject UC7 was critical of the way in which the method introduced and described large numbers of concepts, saying it was 'a lot to digest'. For the readability of materials question, the experienced/ novice differences were large in the case of both methods. Subjects UC9 and UC10 both referred to the bulk of terms and concept descriptions in the accompanying literature.

8.11.2. User Questionnaire

The users in all twenty sessions were given attitude questionnaires with the same 1-7 rating scheme. Both sets were asked seven questions. However, not all the questions were identical for users of both methods. In some cases the methods' approach differed significantly enough to make differently worded questions necessary (e.g. how the period of system use is conducted). The UC method asks the user to take notes where required. The MMA method involves the evaluator prompting the user for descriptions and asking clarification questions. The questions in this phase were tailored to the particular method. The retrospective phase in the MMA method consists of a question and answer session referring to the user's experience of the system, whereas UC users complete the checklist at this point. Therefore MMA subjects are asked about their memory of events when asked, and the UC users about terms and concepts. Both sets were asked if this phase of the session referred (or allowed them to refer) to the important issues.

There were no significant overall differences betwenn ZMMA and UC subjects' answers (P = 0.085, ANOVA). Both sets of users found the package difficult to use (MMA3.2, UC 4.0). Satisfaction with the instructions rated 4.9 for MMA subjects and 4.7 for UC subjects. However, subjects UC7 and UC9 complained that the scenario had misled them, particularly on the status of the lower paragraphs (which where below the screen at the beginning of the task).

User's Review

	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	Average
How easy to use did you find the package?	2	3	3	3	3	4	3	5	3	3	3.2
Did you find the instructions clear?	4	5	5	5	5	6	4	7	4	4	4.9
How easy did you find the task?	2	2	3	2	3	5	4	6	3	3	3.3
How realistic did you feel the task was?	1	5	4	4	7	6	6	5	5	6	4.9
Did you find the evaluator's questions to you during the session were distracting?	3	5	7	6	7	6	6	7	7	7	6.1
Were you generally able to remember things about the session when asked?	6	5	7	4	6	6	7	7	4	7	5.7
Did the questions asked after the session generally referred the important problems?	6	5	6	6	6	7	6	6	7	6	6.2

Figure 8.13a: Satisfaction Ratings by MMA User Subjects

User's Review

	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	Average
How easy to use did you find the package?	4	2	3	4	4	5	4	6	5	3	4.0
Did you find the instructions clear?	7	4	6	5	7	4	2	5	2	5	4.7
How easy did you find the task?	4	3	3	2	5	4	3	6	3	3	3.6
How realistic did you feel the task was?	7	4	7	7	6	4	7	5	6	7	6.0
Did you find that note-taking during the session distracted you?	7	7	7	-	7	-	-	7	7	7	7
Was it easy to comprehend the terms and concepts in the analysis method?	1	2	4	5	5	3	4	2	1	4	3.1
Did the questions asked after the session generally referred the important problems?	3	5	4	2	7	3	5	3	2	4	3.8

Figure 8.13b: Satisfaction Ratings by UC User Subjects

For the question 'how easy did you find the task' most subjects expressed that they found it hard. Subject UC2 claimed that she felt 'on trial' and was embarrassed about the difficulties encountered. All but two of the subjects rated the task four or above for realism. Subject MMA1 and subject UC4 did not offer explanations of their respective ratings of one and two. Nine of the ten MMA subjects were satisfied or relatively satisfied that the interactive protocol did not distract them from the task. Subject MMA1 felt, however, that questions had deflected him from searching for features which were found later.

The UC subjects universally claimed that note-taking was not a distraction, but most reported that they had not taken any notes during the session. Only three out of ten subjects submitted notes at the end of the session. These subjects had only offered brief one word comments which were not comprehensible. Subjects UC2, UC6, and UC8 all claimed that concentrating on the task and scribbling notes were incompatible (and that their scores referred to the fact that they chose not to take notes, and therefore weren't distracted). Subject UC5 claimed that the notes would be no

use as a memory aid for a session of 30 minutes, and that it was not obvious whether anything in particular was worth noting.

Four of the ten MMA subjects claimed they had no problem remembering incidents from the session when asked. The other six rated between four and six. No explicit criticisms of the memory demands were offered.

The average rating of the UC for comprehensibility of terms and concepts was 3.1. Subject UC1 said that 'there is too much jargon to take in'. Subject UC2 claimed she 'hadn't a clue what most of it was asking'. Subject UC6 claimed that 'some of the questions made no sense at all.....you can't tell if they are applicable or not'.

All the MMA subjects rated the after-session debriefing at five or above for relevance to the important issues, with an average of 6.2. Subject MMA9 claimed 'they got straight to the point and allowed me to say what I thought'. However, subject MMA4 claimed 'there were general points I wanted to make which the questions did not ask'.

The Usability Checklist scored an average of 3.8 for relevance to the important issues. Subject UC1 claimed 'you can write what you want eventually, but it is not clear what a lot of the questions are asking'. Subject UC8 claimed 'I knew what I wanted to say, but I did not know what to put for a lot of questions'. Subject UC6 said 'I'd have been happier just filling in the end sections or writing a couple of paragraphs myself'. Subjects UC3 and UC4 both felt that the best part of the checklist was section 11, where written answers were directly requested. Subject UC5 gave a rating of 7, adding 'I think the questions are very relevant but I wasn't good enough to do them justice'.

8.11.3. Summary

A theme that emerges from both the evaluator and user questionnaires is the problem of terms and concepts in the UC method. Six of the ten evaluator subjects, including three with some HCI experience marked the UC method 4 or below for ease of understanding. Also, the users expressed considerable problems, despite the fact that the evaluators offered definitions of checklist questions on request.

Another problem for the UC method seems to be relevance to what the user believes to be the important issues. In particular, the detailed questions in sections 1-8 were criticised. This may partly be explained as being a consequence of the problem of terminology.

8.12 Conclusions

The MMA method scored impressively for effectiveness, efficiency and usability. The effectiveness was demonstrated by the significant difference for the number of suggestions generated. The evaluators identification rate, and production of solution suggestions were highly encouraging. The method users produced a good number of specific recommendations. This suggests that the method contributes to evaluator understanding of incidents. It also suggests that the method is effective in alerting the evaluator to the space of possible design alternatives, encouraging and directing creative thought.

The MMA method scored comparatively well for precision of solutions. This appears to result from the stronger links between observed errors and solution suggestions. The method scores well both for analysis of critical incidents, and references to those incidents in solutions. The MMA method's emphasis on diagnosis rather than classification seems to be effective. This may be due to the fact that relevant error data is captured in detail and analysed. The UC method does not provide the means to such a diagnosis. Critical incident data may be translated into checklist ticks or comments, which may stress particular aspects of it. A single incident may simply be referred to in a checklist as an example of a general trend. This seems to diminish its utility for detailed diagnosis of design failings.

The MMA method also did comparatively well for time taken. The post-analysis session with the user did not take as long as expected, but appeared to produce useful data. Evaluators seemed able to generate solutions reasonably rapidly. The method was designed to avoid wasting time through excessive form-filling, a problem referred to by Wharton et al (1992) in their analysis of Cognitive Walkthroughs. The use of a single form for recording observations, user's responses to interventions, and post hoc analysis, seems to be effective.

The UC method, whilst generating a number of issues and design suggestions, appear to suffer from a lack of focus. The solutions produced were, overall, less specific and less directed towards redesign of features. The studies showed that the large questionnaire tended to make the treatment of critical incidents rather fragmented and imprecise. This may be linked to evidence of evaluator frustration and annoyance at the restriction placed on them by the format.

Presenting user subjects with a large number of human factors concepts is a questionable approach. Users are faced with the task of applying and interpreting

usability criteria, something that has been shown to be difficult even for those with an HCI background (Grudin 1989, Nielsen 1992). Also, asking user subjects to interpret sessions in this format appears to make it harder for them to express their experiences. To allow them to express their personal view of the system seems to be both more acceptable, and capable of yielding more valuable data. The statistics for time taken also suggest that the checklist is an inefficient way of eliciting the user's post hoc view of the system.

The MMA method met with the approval of most evaluator and user subjects. The balance between direction and creative thinking seemed to be acceptable to evaluators. The user subjects seemed happy with the task of recounting experiences with the system. They were able to remember key incidents and describe their experiences in the necessary way. The feeling of control, and enhancement of (rather than intrusion upon) the evaluators' own approach is also important. This may make the method more acceptable to the design community.

Chapter 9 - Summary of Objectives, Developments and Future Work

9.1. Review of overall aims

This thesis had two major objectives. These were to give DM interaction modelling a firmer theoretical footing, and to utilise that theory in developing an accessible evaluation method. This section give an overview of the developments made during the project. The following sections discuss the project's contribution to particular research themes. Section 9.3. will discuss possible further work that can build upon the project's developments.

The Model of Action applied the Theory of Action proposed by Norman (1986) to DM interaction. Although Lewis et al (1990) and Polson et al (1992) have applied this theory to walk-up-and-use interfaces, no attempt had been made to apply it to DM evaluation. The Model of Action linked stages in the cycle of action to types of knowledge recruitment. The model was also used to generate an error taxonomy. This was used to categorise incidents in the subsequent empirical study of MacDraw.

The study of MacDraw combined protocol analysis (Ericsson and Simon 1983) with a model-based data analysis. The three phenomena described in the model (the sequence of user behaviour, knowledge sources, error types) were further investigated. User behaviour was checked for conformance to the cycle of action described in the model. This demonstrated the need for further elaboration of the model. The studies showed that the original account of action specification could usefully be elaborated. Also, this analysis demonstrated the need to separate accounts of expert and novice interaction. The analysis also demonstrated the need for a richer account of the way in which knowledge is recruited by users. This also implied that the error taxonomy proposed in Chapter 3 required expansion to explain breakdowns associated with different types of user activity and knowledge.

Further theoretical work applied concepts drawn from the work on exploratory learning by Lewis (1988) and levels of mental process (Rasmussen 1993) to the DM model. The three levels of processing described by Rasmussen used to distinguish between expert interaction, and sources used in novice learning and interaction. The account of processing levels is elaborated, distinguishing knowledge recruitment for

search and recognition from knowledge recruitment for specifying operations. This provides a rich account of the way in which the system metaphor, and interface organisation effect recruitment from a range of knowledge-spaces. This also accounts for the influence of previous experience of other systems and packages on user behaviour.

The model was modified in the light of the theoretical developments, and used as the basis for an evaluation method. The approach was a variation on the work of Ericsson and Simon (1983) and Wright and Monk (1991). A sequence of abstract interaction types linked to potential usability problems was presented to experimental evaluation subjects. The MMA method guided the collection of relevant data from users in co-operative evaluation. The comparison of the method to Checklist evaluation raised further issues on the emergent theme of evaluation method usability. The MMA method scored well for efficiency, accuracy and relevance of data, and produced more precise solutions. Issues such as locus of control and enhancement of natural evaluator skills were raised.

The analysis of Microsoft Word introduced an evaluative technique developed from the work of Hollnagel (1993). The concept of error phenotypes is modified to incorporate model-based classification. Phenotypes are separated into problems before, during and after an action. Model-based analysis of user intentions, knowledge recruitment and interpretation of the interface is used to establish genotypes, and an agenda for design improvements.

9.2. Developments and Contributions

9.2.1. Contributions to Basic Research

The model of action synthesises work on the action cycle by Norman (1986), a description of processing levels by Rasmussen (1986), and the error phenotype/genotype distinction proposed by Hollnagel (1983). The model provides a rich account of the action cycle, which can account for both internal and external influences on user behaviour. It describes a hybrid user model, incorporating users' knowledge of the task-space, along with their interpretation of system rules and principles gained during interaction. Distinct reasoning levels are described with reference to the content of user's models. This knowledge can be employed in the interpretation of user problem phenotypes.

The Model of Action develops the Theory of Action proposed by Norman (1986) to describe display-based action. The model is elaborated to account for mental acts involved in feature search and recognition, identifying critical steps in DM dialogues. This accommodates the claim that search for features and specification of operations may be seen as distinct mental acts. This distinction is not made explicit in Norman's Theory of Action. The model includes a description of user responses to errors. This develops a theme which is not addressed by established information processing models, such as the Model Human Processor proposed by Card et al (1983).

Rasmussen's (1986) three layer description of reasoning is applied in the model, distinguishing novice, intermediate and expert interaction. The levels are linked to knowledge-spaces utilised in interaction. Knowledge recruitment from the task-space is separated from generalised knowledge of the system. Knowledge-based processing is described as interaction prompted by the user interpreting the system image in terms of concepts from the task-domain. In particular, knowledge-based processing involves the interpretation of on-screen metaphors and affordances using heuristics and principles from general task and domain knowledge. The model unpacks the role of metaphor in DM designs. The knowledge-based level distinguishes between the the interpretation of spatial and organisational metaphor (for search), metaphor in feature identification, and action metaphor cueing operations.

The model of rule-based processing applies the work of Lewis (1988) on example-based learning to explain the role of spatial and episodic feature memory in system learning. The account of example-based learning is expanded to include operational knowledge acquired from interaction with other packages, familiar features from previously used packages, and packages with a similar look and feel. Use of novel features is described in terms of a set of possible sequences of mental acts. Rule types employed by users are distinguished by the mental acts with which they are associated (i.e. general rules, recognition rules, operational rules, interpretation rules). These mental acts may require distinct types of system support, with the user recruiting from different knowledge-spaces for each act. Skill-based interaction is described as the automatic triggering and use of compiled procedures in interaction. This level of processing is likely to feature in most interactive sessions as users learn and repeatedly apply basic rules of operation.

The model incorporates themes of internal/external task mapping described by Moran (1983) and Payne (1991). The influence of task-space knowledge on system metaphor interpretation, feature recognition, and choice of operations is described. The model treats these three uses of external knowledge as separate, affecting distinct aspects of user performance. In doing so, it provides an account of schema extension

in learning, with particular reference to the role of task/domain knowledge. This aspect of the work also addresses issues about the nature and extent of metaphor influence on user behaviour, a theme discussed by Laurel (1988) and others. The model uses notions both of example-based learning within the system, and the influence of the device independent task-model. The empirical studies suggest that this hybrid account may be required for an adequate interpretation of user behaviour in interactive sessions.

The use of metaphor is distinguished by the type of external knowledge recruited, and the user mental acts that it supports. The role of system metaphor in the large is spatial and organisational. This links to navigation, and selection of search locations by novices. It subsequently has a role in supporting rule-based search in which the user links areas to types of feature. Also, users assign operation types to distinct feature groupings. In this sense metaphor establishes examples from which generalisations may be made, providing an initial semantic link between task-space and device-space. A distinction is drawn between system metaphor and individual feature metaphors. Feature-level metaphor may have solely a 'feature identifier' role, or present an action metaphor to the user. The latter is effective where the general system metaphor is unable to express needed operations, or where there is no mapping between task-space and device-space for a particular function.

9.2.2. The MMA Method as a Contribution to Evaluation Techniques

Various taxonomies have been used in contemporary evaluation approaches. Booth (1990) uses a four element taxonomy for the classification of errors. Nielsen (1993) and Ravden and Johnson (1989) use taxonomies of usability principles. These methods are now compared to the MMA model described in the thesis.

Booth (1990) uses the concept of user-system mismatches, with the evaluator prompted to identify either an object or operation (on objects) that is problematic. Having distinguished this element, the evaluator is prompted to identify either a concept mismatch (a mental or computational representation of an object or operation), or a symbol mismatch (a token representing an object or operation). The taxonomy is used in a similar way to MMA, with incidents analysed to find the root cause of a dialogue failure. However, the taxonomy is relatively limited. No distinction is made between user problems prior to an action, and problems interpreting the result of action. MMA, by contrast, is explicit in linking its taxonomy to the cycle of interaction.

Nielsen (1992) uses selected usability principles and guidelines as a taxonomy for

Heuristic Evaluation. The selected heuristics, it is claimed, broadly cover the range of possible usability problems. Nielsen (1990) selects a number of examples which clearly fit the cited heuristic categories. However, the strength of evaluation must surely be its completeness in identifying and remedying design errors. Indeed Nielsen's further studies (Nielsen 1992) show that Heuristic Evaluation can be far from complete in trapping errors, although further testing(Nielsen and Phillips 1993) showed the approach is a comparatively efficient way of conducting evaluation. Evaluators are given categories with which to search the system. Therefore, they may find problems which seem superficially to fit one or more category. However, deeper analysis of the system is left to their own inspiration, a problem that makes it difficult for non-HCI experts to use (see Dutt et al 1994).

A further problem with Heuristic Evaluation was discussed in Chapter 2. It may be unclear how to interpret errors in terms of the taxonomy. Consistency, for example, is a term that potentially bears diverse definitions and interpretations (see Grudin 1989, Reisner 1990). The MMA method addresses both of these problems. The MMA method acknowledges the fact the the surface appearance of a design problem may give only a clue as to the deeper problem. Also, the taxonomy is linked to the elicitation of protocol evidence, rather than the interpretation of heuristics and principles.

Existing dialogue principles fall into four categories when examined in terms of canonical interaction models. These are principles that are relevant (e.g. provide feedback), redundant principles (e.g. error messages where a system has none), unfocussed principles and complimentary guidelines (e.g. avoid cluttered screens). Some dialogue design principles seem redundant when examining DM systems. For example, 'provide good error messages' seems to be subsumed within 'provide feedback'. The provide feedback principle is shown to be more focused, as it accurately refers to a component of the interaction cycle.

The problem of unfocussed principles is demonstrated by comparison with the model. Some very general principles such as 'be consistent', 'simple and natural dialogue' and 'speak the users language' touch upon diverse elements of design. The diverse nature of action-cycle steps and dialogue types is not reflected in these principles, although Nielsen (1993) attempts to redefine them for application to modern interaction styles. The model-based analyses in the thesis suggest that linking usability principles to the action cycle could provide better focus. Dutt et al (1994) report studies of subjects using Nielsen's (1992) Heuristic Evaluation method. The study revealed some problems understanding and applying some of the heuristics (principles). These included the 'consistency', 'simple and natural dialogue' and speak the user's

language'. This concurs with the analysis in Chapter 2 of the thesis, that the focus of these principles may be too broad or vague to assist novice evaluators. Dutt et al (1994) also noted that some heuristics (e.g. minimise memory load, good error messages) were largely redundant for pinpointing errors, a point also made in Chapter 2.

The Usability Checklist method (Ravden and Johnson 1989) attempts a comprehensive taxonomy of usability issues, citing 161 aspects of usability. The Chapter 8 study demonstrated that this approach adds complexity to evaluation, without overcoming the problem of accurate classification. The study demonstrated that usability problems may (potentially) be classified in terms of several principles. This is compounded by the absence of criteria for determining the actual effect of a particular design element. The cited criteria give only a vague hint to evaluators about error causes and appropriate redesign.

The problems discussed above highlight some important weaknesses of taxonomies. One is the fact that taxonomies, by their very nature, make strong claims about coverage. An 'ideal' taxonomy covers every single relevant phenomenon and example in the domain with which it deals. Furthermore, it divides and categorises phenomena according to the most significant concepts and descriptions. Small taxonomies may, therefore, carry too broad a set of categories. Conversely, detailed taxonomies, of which Ravden and Johnson (1989) provide an example, make so many distinctions that accurate classification becomes difficult. Therefore, criteria for applying taxonomies is required. The Model of the Action cycle provides such criteria. The evaluator is given both a taxonomy, and techniques for assigning examples to categories.

The cited problems of taxonomy focus, relevance and redundancy are addressed by the MMA approach. The focus problem is dealt with by using category assignment within the context of a model of action. Evaluators identify the relevant point in the action cycle and the relevant interface element or feature. Therefore different manifestations of the 'consistency' problem are handled (e.g. consistency of representations, consistency of feature behaviour). Similarly, the process avoids focus on redundant or irrelevant categories, by helping evaluators to pinpoint relevant criteria. Also, the method, because it pinpoints the nature of current usability problems, can be used in harness with some of the specific usability principles which form part of the Usability Checklist taxonomy. For example, once problems with salience or discriminability of feature cues are isolated, more specific guidelines on use of presentation techniques (e.g. ISO 9421, Part 2) may be applied.

The MMA method uses a Model of Action, linked to errors, to interpret user protocol data. As discussed above, model-based analysis is required for the effective application of usability criteria. Similarly, this type of analysis is needed to establish a real understanding of user-based evaluation data. Methods such as protocol analysis (Ericsson and Simon 1983) and the York Manual (Wright et al 1991) provide guidelines for data collection from user studies, but do not provide criteria for their interpretation. The Chapter 7 study shows that the distance between a perceived problem (error phenotype) and the root cause of a problem (genotype) may be considerable. This claim is backed by Carroll et al (1993) who make the distinction between critical incidents and critical threads in describing distant error causes. The MMA method provides guidelines for both data collection and interpretation.

The MMA Model of Action is used to focus attention on usability issues. In this sense there is a similarity between the model and the Cognitive Walkthrough technique (Lewis et al 1990). The Cognitive Walkthrough technique is a step-by-step evaluation of a design. Whilst action sequences are a central part of this technique, the models used may not be sufficiently detailed to capture the range of error types associated with DM interfaces. The MMA method specifically uses roles, linked to stages of an action, to focus usability questions. This allows for a more detailed analysis of potential problems. For example, Polson et al (1992) describe a questionnaire for the step of 'choosing and executing an action'. This includes questions directing evaluators to the availability of the action, the 'action label', its link to to the goal, and the possibility of wrong choices. This analysis does not explicitly distinguish between locator problems and feature identifier problems. Also, the questionnaire implicitly assumes that if users link a feature label to a goal, they will be able to perform an action. However, the model analysis in Chapter 5 shows that a number of the cited issues require a more detailed analysis to establish the real cause of a problem. It is useful to trace causes of missed features either to the design of the feature (feature identifier) or the visual metaphor that leads users to features (locator). Also, identification of features does not necessarily imply that users can work out their operation (operation specifier). The use of roles to analyse incidents in MMA facilitates a more precise diagnosis of the causes of user problems. Whilst the cognitive walkthrough (Lewis et al 1990, Polson et al 1992) is a useful tool for predicting error phenotypes, it only partly provides help in establishing genotypes.

Another contemporary approach to evaluation is Claims Analysis (Carroll and Campbell (1989), Carroll and Kellogg (1989)). This proposes the analysis and refinement of existing designs, using claims (lessons) from their study to inform new designs. It is claimed by Carroll (1990) that this technique replaces cognitive theory as as support for evaluation and design. However, this claim is disputed by among

others Polson et al (1992), who argue that the interpretation of designs using cognitive theory is necessary for Claims Analysis process.

The MMA method provides a technique for interpreting design problems as cognitive claims. The success or failure of certain interface techniques may be assessed by diagnosis of user problems. Menu names and icons embody claims that the user will have equivalent representations of the task-space. A relatively complex feature operation will also make claims about the user's ability to use abstraction-based generalisation, or metaphor comprehension to work it out.

9.2.3. MMA as a Contribution to Practical Evaluation

The utility of user-based evaluation for DM interfaces has been illustrated by the project. The method has the advantage of gathering information about how the display effects user expectations. It also accounts for learnability of system functionality through example-based learning.

The study in Chapter 8 suggests that evaluation techniques based on models are more appropriate on several counts than structured human factors questionnaires. The model-based method (MMA) scored favourably for effectiveness, efficiency and user satisfaction. The effectiveness of the method was demonstrated by the greater number of explicit solutions offered by the method. The time taken to produce the solutions also compared favourably with the Usability Checklist. In particular, participants expressed dissatisfaction at the length of time taken to fill in the usability checklist. User satisfaction was also generally better for the MMA method.

The study raised a number of issues on the theme of method usability. The principle of effectiveness in pinpointing user problems has been the crucial concern of previous evaluation techniques (e.g. Karat et al 1992). This was extended to effectiveness in producing design solutions. Hitherto, analysis had focused on identification and diagnosis of usability problems (e.g. Jeffries et al 1991). The study showed that the Usability Checklist method did not produce information that focused on individual design problems. One the other hand, the MMA method was geared to interpretation and comprehension of critical problems. This approach showed encouraging results when used by experimental subjects. The method proved effective in helping evaluators to navigate the space of possible solutions. This requirement for evaluation methods is noted by Booth and Gray (1990).

The UC sessions in the Chapter 8 subject demonstrated the problem of linking analysis of users' problems to design improvements. The UC evaluator subjects collected a

considerable volume of information. However, statistics showed that focus on issues, and links to critical incidents were often vague. The study suggested that the usability checklist prevented the user participant from expressing some of what they wanted to say about the session. Also, the format was reported as restrictive by most of the evaluator subjects. The problems experienced by the subjects demonstrate the problem of collecting appropriately focused data.

The themes of efficiency and user satisfaction are tackled in other reports on evaluation methods. Wharton et al (1992) report considerable difficulty in applying a Cognitive Walkthrough to advanced, functionality rich systems. Not only was filling in the forms found to be inefficient, but it also provoked complaints from users in experimental sessions. In particular, the repetitive filling-in of forms was unpopular with subjects. The MMA method limits form filling to gathering data on critical incidents. In ten sessions the number of incidents was approximately nine per session. This minimises the amount of note-taking that is required.

A number of reports on contemporary evaluation methods have emphasised issues of method comprehension. One theme is the amount of expertise required to perform an evaluation. Nielsen (1992) claimed that any Heuristic Evaluation team should consist of one or more human factors experts. It is also claimed (Wharton et al 1992) that the cognitive walkthrough method requires a Human Factors (or cognitive science) expert in a small evaluation team. Therefore, these methods would only be of use in an organisation if someone suitable is available. Otherwise, a consultant could be hired, at considerable cost. The MMA method, however, was learned and used to good effect without the personal involvement of a human factors expert. The use of 'roles' provided a common and relatively simple way of analysing a range of reported problems.

Dutt et al (1994) cast doubt on the ability of Walkthrough methods and Heuristic Evaluation to make effective redesign requirements. In particular, they point to the lack of access to user goals and characteristics. The MMA method addresses this problem by eliciting users' models of tasks as they are performing them. Dutt et al (1994) recommend the use of task analysis methods (e.g. Diaper 1990, Johnson et al 1988, Johnson and Johnson 1991) to identify requirements for redesign. The MMA method elicits user models at the point when the display is an influence. This may be more effective than task analysis which examines device independent task-models. Theoretical work by Lewis (1988) suggesting that users generalise procedures from single examples was supported by evidence from the Chapter 4 and Chapter 7 studies. The chapter 7 study showed an example (i.e. user expectations of the column-making operation) in which user's expectations were formed from example-based

generalisation, whereas the system model bore more resemblance to their device independent models of the task. This demonstrates the need to elicit 'situated user models' which account for the influence of the current display and recent interaction, as well as external task models.

The empirical studies described in Chapter 6 also propose and assess techniques for analysis of user errors in co-operative sessions. The goal-tree analysis shows the model the user holds is a synthesis of concepts from the task-space and the device space. The analysis is capable of revealing not only the influence of task concepts, but also inferences drawn from previous interaction. This method of eliciting user models therefore provides information which is not provided by task analysis.

The goal-tree analysis derived from the work of Keiras and Polson (1985) was successfully augmented with elicitation of the user's view of the system state. This analysis reveals misconceptions such as failure to perceive or comprehend previous state-changes, or confusion caused by the system metaphor. This questioning can reveal distant causes of errors.

The 'constrained search test' technique helps diagnosis of users' failure to find features. The cause may not be clear from normal protocol data. If the user can recognise a feature from an array, the cause of the problem is more likely to lie in higher level aspects of design. Also, 'motivation analysis' can help determine the misconceptions behind an incorrect choice of feature.

The combination of techniques allows a rich description of users' models in the presence of the device. This provides information which situates analysis of individual task performance and feature use within the user's overall model of the system, and beyond to their knowledge of other packages and knowledge of the domain.

The method was designed for use by novice evaluators. The Chapter 8 study suggests that it is suitable for novice use, although this study did not involve studying industrial evaluation in context. The study suggests that it more accessible than other contemporary methods, including Heuristic Evaluation (Nielsen 1992), and the Cognitive Walkthrough (Polson et al 1990) which require the participation of experts. The Chapter 8 study shows that the MMA method is capable of finding and diagnosing errors. Unlike the methods cited above, MMA provides for analysis which produces solutions. Other methods (e.g. Wright and Monk 1991) generate data without directly assisting in its interpretation.

9.2.4. The Model-Based Approach and Contemporary Evaluation Practice

This section compares the MMA method to approaches currently used in non-academic organisations. Contemporary research developments, such as Heuristic Evaluation (Nielsen 1993) and the Cognitive Walkthrough (Polson et al 1992) are not widely employed in industrial evaluation settings. Enquiries conducted on evaluation practice at Logica PLC and British Telecom PLC and with industrially-linked HCI experts suggest that method acceptance industry is still rare. The state-of-the-art in academic evaluation methods is not reflected in commercial practice.

Enquiries suggest that both detailed guidelines and empirical testing are used in industry, though with little intersection between them. Guidelines include ISO standards, e.g. ISO 9241 part 16 which cover DM style interfaces. These are similar in content to traditional human factors principles, and the Usability Checklists by Ravden and Johnson (1989). Another variant of this is the Style Guide. British Telecom, for example developed their own Style Guide, citing conventions for the appropriate look and feel of their own interface developments. Usability Metrics are used to assess interface performance. Number of usability errors and performance time are examples of the counts taken.

Formative methods such as GOMS (Card et al 1983) have had limited industrial application, although GOMS analysis has successfully been applied to telephone toll assistant services (Gray et al 1990). In this example the method demonstrated the possibility of reducing performance time by refining task design.

Expert usability reviews of in-house developments are used by Logica PLC and others. However, the nature of the expertise involved in such evaluation is diverse. Some companies may have a resident HCI expert at their disposal, while other organisations use expert designers and domain experts in early evaluation. Logica uses this approach, and employ scenario-based user testing in later stages of the design process. The expert reviews are not structured, and simply elicit expert opinions.

A number of organisations (e.g. Nat West PLC, British Telecom) have their own usability laboratories. Usability tests involving users are widely employed, but there is little evidence of structured methods being employed. Some organisations (e.g. Lloyds Register of Shipping) use guidelines for user testing developed by internal Human Factors professionals. However, there remains a marked absence of method-based analysis of data generated from usability studies. Other user-based testing includes the use of usability questionnaires (see Olphert 1986). Both experts and

potential users are invited to give a rating for selected aspects of a product's usability. These rating are then totalled to establish the system's strengths and weaknesses. The HUFIT project (see Olphert 1986) used user ratings, along with measures of error rates and task completion scores to assess products.

An alternative approach used by DEC (see Whiteside et al 1988) is Contextual Evaluation. This approach emphases the evaluation of products in the actual workplace. It is claimed that usability laboratories provide a synthetic and therefore misleading context, which may render any evaluation findings inaccurate or incomplete. In particular, the users are performing tasks set by an experimenter. Contextual Evaluation assesses users performing tasks relevant to their organisation and career (Whiteside et al 1988). What is less clear, however, is the scope for actual method use. Whiteside et al (1988) refer to a suite of performance monitoring, benchmarking tests, surveys and questionnaires, but give little detail of these. The finding and addressing of usability bugs by Whiteside et al (1988) appears largely reliant on feedback from surveyed users.

Co-operative Evaluation has been successfully applied to industrial products evaluated in context (Rowley 1994). Rowley (1994) describes this as bringing the usability laboratory into the workplace. However, this method is primarily a procedure for data capture and recording usability problems, and does not offer diagnostic advice. The MUSIC project, which provides performance metrics for testing usability, also addresses the issue of context (Maissal et al 1991). The MUSIC approach draws conclusions about usability from gross statistical counts. Some contextual variables cited by the authors are representativeness of users, of tasks and the naturalness of the working environment. Questionnaires are used to establish the representativeness of users.

The picture in industry suggests that there is scope for greater use of method in industrial evaluation . Expert evaluation seems to be generally unstructured. There is potential for use of diagnostic aids (which Heuristic Evaluation tries to address), as well as methods for interpreting collated findings of a number of experts in one product evaluation. Similarly, there is potential for augmenting guidelines with advice for their interpretation and application to usability problems. Data from user-based studies requires considerable interpretation, and could benefit from method-based analysis. Generally speaking, the employment of evaluation approaches is dependent on the physical and human resources that are available. Methods and techniques may help to maximise the effectiveness of both. In the former case, methods can make more efficient and effective use of laboratory facilities and usability data. In the latter case, levels and types of evaluator expertise may be enhanced by appropriately targeted

methods. The following paragraphs relate the theory and techniques to contemporary practice within organisations.

Expert evaluation is usually conducted without use of methods. Also, an 'expert' in this context could be an HCI expert, a design expert with appropriate experience for the type of system being scrutinised, or a domain expert. The MMA models provide a 'tool for thought' which provides assistance both on initial investigation of products and a clearer interpretation of redesign needs. The models emphasise a stepthrough evaluation linked to the interaction structure of the system. The evaluator is primed to consider interface design features, user's skill and knowledge types in the interpretation of problems. However, the expert is still responsible for identifying problematic features and suggesting solutions. Therefore, a tool for thought may provide an acceptable level of help which improves the process without infringing the expert's own way of conducting evaluations. Such a claim would require testing and validation using experts.

Design and domain experts may lack a degree of relevant HCI knowledge. Therefore, they may require assistance in sharpening and focusing their interpretation of problems. Whilst they may be able to use their own knowledge and experience to identify sub-optimal design, they may be less able to characterise the problem, or reason from observed problems to an appropriate re-design. The MMA approach encourages both the identification of a broad range of problems and their interpretation in terms of user (domain) knowledge as well as more general HCI knowledge. Also, it provides a format and technique which can be used to link problem characterisation to descriptions of potential solutions.

The common use of user-based testing in industry was a major motivation for this research and its development of model-based tools for interpretation. Reports suggest that it is still common for 'method' in these evaluations to simply mean instructions for setting up and conducting the session. Typically, videoed protocol studies produce an enormous amount of data which may be difficult to sift through and to interpret. This is a problem for organisations as it is a burden on time and resources. Therefore, it is helpful for methods to provide ways of focusing attention to cut analysis time, and to extract meaningful conclusions from physical and verbal evidence. The MMA method addresses both problems. The evaluators are encouraged to focus solely on critical incidents, and be guided towards an in-depth coverage of usability problems.

The issue of contextual factors in evaluation was only partly addressed in the thesis. For example, the design of scenarios deliberately mimicked typical tasks with such tools. However, the laboratory setting prevents exploration of the work setting as a

factor. Nonetheless, some of the issues raised by the Contextual Evaluation approach can be addressed using concepts from the MMA method. Complete interpretations of user behaviour require knowledge of their educational and professional background, the types of system they have used, and their objectives in using a system. The MMA method stresses that these factors are taken into account when interpreting critical incidents. The deep traces of error causes include establishing the state of user knowledge, which includes prior computing knowledge and general work experience.

9.3. Future work

9.3.1. Formative Evaluation

The model was tested in a format for summative evaluation, using a working system. However, the model has not been applied to real working environments, or subject to the constraints and difficulties that such environments may pose. A survey of Human Factors practice in industry by Dillon et al (1993) suggests that practical use of the method may be difficult in some organisations. For example, 27% of organisations reported that access to the user population was difficult. This would make model-based user studies difficult to perform. In 51% of organisations evaluation was left to the designer to perform. The authors also observe that interpretation of usability data is heavily skill-dependent, a problem which the MMA method addresses. Therefore, it is desirable to consider its possible application at different points in the design process.

The cycle of action on which the MMA method is based is similar to that which underlies the cognitive walkthrough technique (Polson et al 1990). The Cognitive Walkthrough generates design questions in a continuous sequence which reflect paths of user action. The model developed in the thesis affords a similar style of analysis. The stages of the prescriptive model can provide basic format for a future walkthrough technique.

For a model-based walkthrough to work for DM interfaces, two elements would need to be present. These are identification of dialogue types and user task-model information. Chapter 5 emphasises the need for distinguishing between dialogue types, with particular regard to the type of mental strategy that is triggered by the interface. Let us use the example of walking through a sequence including a palette selection action. When the 'specify operation' phase is examined, an estimation of the user's previous interaction experience or trained knowledge must be made (some basic training on interface functions can be assumed). The current feature may be examined

for its resemblance to the generalised knowledge of the palette that the user has. In this case (unlike, for example, a scroll-bar) the user is able and likely to recruit generalised knowledge of palette operations. This will prompt the user to try a similar operation. The design can thus be checked for predictability of operation. In the counter-example, the scroll-bar gives the user the task of working its use out without the help of rule-based generalisation. Accordingly, assessment of learnability throws greater emphasis onto adequacy of metaphor suggestion, and feedback for exploratory action.

The second requirement is that a relevant assessment is generated of the external knowledge that the user will bring to the task. This includes typical pictures of the domain, of feature names, and of object behaviour and relations. The objective is to collect canonical models of typical users' knowledge. Some elicitation methods have been reported in the literature (see Diaper 1990). The user information needs described by the model indicate the type of knowledge that would be in such a model.

Contemporary evaluation methods stress the need for user involvement rather than exclusively relying on estimates of the target user. Another stated requirement is that evaluation be integrated with design practice at an early stage (Dillon et al 1993, Shackel 1991, Whiteside et al 1988) . This is envisaged as involving simulations and prototypes, along with user and task analysis. Usability walkthroughs have been designed in order to support testing early in the design cycle (e.g. Lewis et al 1990, Karat et al 1992).

The MMA method is aimed at evaluation of running systems or prototypes. However, the user model on which it is based could be developed into a walkthrough format. The set of criteria attached to activities and roles (described in chapters 5 and 6) may be used to test factors such as predictability, learnability and feedback design on a prototype.

The Model of Interaction is also pertinent to the wider theme of integrating system and user models for formative use in design. Barnard and Harrison (1989) describe the problem of reconciling formal descriptions of systems with psychological models of users. The Model of Action encourages designers to understand user-system dialogues as sequences of system supported mental acts. The model's description of the action cycle could be used in an integrated description of a system which links system acts with 'likely' user responses. Such an approach could contribute to the integration of human factors in the design process.

There is also further scope for using roles in developing object-orientated design.

Object-orientated user interface design involves phases of listing typical objects associated with users, selecting which objects will be visible at the interface, and describing object appearance and relations in terms of the system metaphor (Macaulay 1995). The model may contribute structure to description of objects and assist in selection of their representation and operation at the interface. The model provides a sequence of 'properties' such as location, identity metaphor, operational metaphor and interpretation metaphor which may be applied in this process.

The Role-based criteria in MMA could also be applied to the Heuristic Evaluation approach. Studies by Jeffries et al (1991) Nielsen and Phillips (1993) and Dutt et al (1994) suggest that Heuristic Evaluation has some advantages over more structured usability walkthroughs. Particular advantages are cost, learnability and efficiency. Therefore, Heuristic Evaluation may be useful for DM systems. However, the findings of the project are that many of the principles selected by Nielsen and Molich (1990) may lack sufficient focus to effectively assist problem identification. There is nevertheless, potential for heuristic evaluation specifically targeted towards DM systems. Heuristics could be targeted towards roles to assist evaluator focus. The 'provide feedback' heuristic, for example, is already targeted to the evaluation phase of the action cycle. This is linked with visibility and accurate representation of the system state.The analysis in chapter 5 links critical issues to other roles in the cycle of action. These could be presented as heuristics linked to stages of an action, providing an alternative evaluation format.

9.3.2. Evaluation of the Design Process

The theme of tracing errors to source is referred to by Carroll et al (1993). They use an example of an error traced to the training materials given to a subject, and the comprehension of a particular term. The MMA method traces a variety of design problems using user errors as evidence, including assumptions from previous actions, failure to understand names, and metaphor comprehension. If these causes are taken further they may be traced to the parts of the design process from which they originated. For example, an error of feature recognition could be traceable to knowledge capture practice. The task analysis or requirements capture phase may have failed to give accurate information about how task objects are represented. Inappropriate functionality problems (e.g. where the scope of a feature's utility is inadequate) may result from lack of detail in requirements capture.

Future work could link evaluation both to improvements in design and the design

process. Much recent attention has centred on the need to question and evaluate design decisions (MacLean et al 1990). However, this work concentrates on questioning particular design decisions, and providing techniques for this purpose. Evaluation data could be used to expose weaknesses in design approaches by linking errors with relevant parts of the development process. Also, it could address design practice within organisations. Evaluation data is a useful benchmark test for design practice. An organisation may have its own design practice, or combination of techniques which are favoured as company practice. Alternatively, the approach may be ad hoc, decided in design meetings as the design project commences. A summative evaluation of the practices used would help to refine practice and eliminate mistakes in the process. Product evaluation data of the sort collected by the MMA method could have a role in this process. Design problems that are manifest in the product's use may help identify flaws in the process that created them, setting an agenda for alterations. Future extensions of the MMA method could include methods for helping organisations identify flaws in the design process.

9.3.3. Application of the Model to other types of System

The model is designed specifically for use in evaluating DM interfaces. However, DM dialogues may be seen as relatively rich display-based interfaces. The principles behind the method are applicable to other display-interfaces which are not as reliant on a DM component, or a visual metaphor. The description of the sequence of action is valid for other types of system (e.g. Lewis et al 1990). Also, the notions of task-performance, learning by examples, display-guided action, and providing feedback apply to other types of system. For example, information retrieval tasks may be seen in terms of matching task goals to the device-space, identifying features choosing operations and generating further goals from feedback. The underlying principles in the design are broadly similar to the systems that the project concentrates on. The difference is that DM use a variety of visual interactive techniques. However, the account of the sequence and interface roles is similar.

The model could be adapted for the evaluation of virtual reality interfaces. Virtual reality facilitates more sophisticated interaction and a range of task support possibilities beyond conventional interfaces. For example, 'augmented reality' (Adam 1993) allows a combination of real and virtual worlds. Therefore, the system image may prompt direct action in the real world, as well as the virtual world. Also, users can perform a real task whilst observing a system display of the desired task end-state. Also, virtual reality may be immersive, with users wearing datagloves and headsets (DeFanti et al 1993). They may also have images of themselves on-screen. This type

of system potentially allows a more direct transfer of task knowledge and skill than DM. However, it is similar to DM in that the system uses literal domain representations and virtual actions. The Model of Action therefore contains relevant concepts which could be applied to such systems. Greater emphasis could be placed on the transfer of manipulation skills, a central theme in virtual reality.

Some interface techniques have been developed to address problems in novice interaction with DM interfaces. In particular, the problems of identifying features and specifying their use have received attention. Demonstrational icons are an example of this phenomenon (Baecker et al 1991). These icons give animated demonstrations of a feature's utility when requested by users. Baecker et al (1991) found that they were effective in helping to convey the meaning of a tool in cases where the static icon had not been understood. The Word for Windows interface has an online explanation bar at the bottom of the screen which gives textual explanations of menu options. Whilst the potential utility of such features for enhancing user understanding is considerable, there are potential problems with them. These techniques may themselves cause user problems. The danger of confusion between cues is still present with demonstrational icons. Also, it may be hard to convey the full range of a tools functionality.

A number of modern systems allow input and communication for multiple participants. Video-conferencing, electronic meeting rooms allow multi-user dialogues and multi-user task environments (Weiser 1991, Preece 1993). These present problems that can be comprehended in terms of the model. The model refers to user and system as dialogue participants with the system playing information roles. Groupware alters the nature of system output. The user has to be informed of state-changes that were caused by other users. Therefore, the system must give rich feedback on the state of the system and the causes of that state. A number of possible technologies and techniques may be useful in this process. However, their utility may be judged using similar criteria to other display-based dialogues. Therefore, the model could, by using a wider account of system feedback, be modified to handle evaluation of groupware interfaces.

CSCW provides for some potential developments in the design process. The process of knowledge capture faces a number of practical difficulties (Berry and Broadbent 1986, Cullen et al 1987, Cullen and Bryman 1988). One problem is that it is time consuming to go to a number of individuals in an organisation to collect information. The practice is blighted by possible non-cooperation by individuals and organisations. Therefore, a quick and reliable method of eliciting generic models of target users is important. Collaborative prototyping using groupware may generate and evaluate design options rapidly, with the direct input of representative users. The Model of

Action may be used in a step-by-step analysis of design options and requirements. Conflicts of interest can be aired, compromised and resolved in such sessions. Consensus on meaning and representation could also be reached in these sessions.

9.4. Summary

The aim of the project was to develop user modelling for DM evaluation. This was approached by integrating models of user action with error classifications and an account of user processing levels. The Theory of Action proposed by Norman (1986) was specifically applied to DM in a model of user action. The Model of Action expanded Norman's theory, accounting for separate search and specification phases. The Model of Action was expanded further to account for user behaviour in response to errors. The description of user processing levels by Rasmussen (1986) was applied to user recruitment of knowledge under prompting from interface features. This facilitated a rich description of the types of error that sub-optimal DM design may cause. The linking of design errors to users problems utilised the phenotype/genotype distinction described by Hollnagel (1993).

The Model of Action provided the theoretical basis for the MMA evaluation method. The method exploited the use of verbal reports as data, building upon the work of Ericsson and Simon (1984) and Monk et al (1991). The method facilitated the elicitation and analysis of concurrent user models (as opposed to their device independent models). This addressed the complex relationship between external knowledge and the example-based learning of systems. The method was aimed at providing diagnostic aids for novice evaluators. The method aimed to give evaluators a sense of 'the solution space'. Empirical testing of the method suggested that the method helped novice evaluators to achieve this. The study of Microsoft Word also showed the potential of a suite of retrospective elicitation techniques for establishing genotypes.

The findings of the studies suggest that user-based studies may be an efficient way of providing accurate diagnosis and solution of design problems. The elicitation of users' models was demonstrated to be more efficient and effective than the Usability Checklist approach proposed by Ravden and Johnson (1989). Also, the use of models in interpreting problems seemed effective in helping evaluators comprehend

design problems. The MMA approach provided both a method for efficient user model elicitation, and facilitated the interpretation of user errors as evidence for interface design problems.

References

Adam, J. (1993), 'Virtual Reality is for Real', IEE Spectrum, **30**, 10, 22-29

Anderson, J. (1983), 'The Archietechture of Cognition', Harvard University Press

Anderson, J. and Thompson, R. (1986), 'Use of Analogy in a Production Systems Architechture', Illinois Workshop on Similarity and Analogy, Champaign-Urbana, Illinois

Baecker, R. Small, I. and Mander, R. (1991), 'Bringing Icons to Life', in *proc. CHI 91: Reaching Through Technology*, Robertson, S. Olson, M. and Olson, J (eds.) ACM Press, 1-6

Barnard, P. and Harrison, M. (1989), ' Integrating Cognitive and System Models in Human-Computer Interaction' in *People and Computers V.*, A G Sutcliffe and L Macaulay (eds); Cambridge University Press, 87-103

Bennett, J. Lorch, D. Keiras, D. and Polson, P. (1987), 'Developing a User Interface Technology for Use in Industry', in *Proc. Interact 87*, Amsterdam: Elsevier Science Publishers

Bellotti, V. (1988), Implications of Current Design Practice for the Use of HCI Techniques, in: *People and Computers IV*, D.M. Jones and R. Winder (eds.), Cambridge University Press, 13-34

Bellotti, V. (1990), 'A Framework of Assessing the Suitability of HCI Techniques', *Proceedings Interact 90*, D.Diaper et al (eds), Elsevier North-Holland, 213-218

Berry, D. and Broadbent, D. (1986), 'Expert Systems and the Man-Machine Interface' Expert Systems, **4**, 3, 152-168

Bewley, W. Roberts, T. Schroit, D and Verplank, L. (1983), 'Human Factors Testing in the Design of Xerox's 8010 'Star' Office Workstation', *Proceedings CHI-83*, A Janda (ed), ACM press, 72-77.

Booth P.A. (1990), 'ECM: A Scheme for Analysing Human-System errors', *Proceedings Interact 90*, D.Diaper et al (eds), Elsevier North-Holland, 175-181.

Booth P. A and Gray J., (1990), 'Errors and theory in human computer interaction',

Acta Psychologica, **78**, 69-97

Brennan S.E., (1990), 'Conversation as Direct Manipulation', in *The Art of Human Computer Interface Design*, B. Laurel (ed) Addison Wesley, Reading MA

Browne, D. (1988), 'Human Computer Interface Design Guidelines', Ablex, NY

Bullock, M. Gelmen, R. and Baillargeon, R. (1982), 'The Development of Causal Reasoning', In *The Developmental Psychology of Time*, Friedman, W. (ed.) New York: Academic Press

Card, S. Moran T. and Newell A. (1981), The keystroke level model for user performance time with interactive systems, *CACM* **23** (7), 396-410

Card S. Moran T., Newell A, (1983), The Psychology of Human-Computer Interaction, Laurence Erlbaum Associates

Card, S. Robertson, S. and MacKinlay, J. (1991), The Information Visualiser: An Information Workspace, *proc. CHI 91: Reaching Through Technology*, Robertson, S. Olson, M. and Olson, J (eds.) ACM Press, 180-188

Carroll, J.M. (1984), 'Minimalist Design for Active Users' in *INTERACT 84 - The First IFIP Conference on Human-Computer Interaction*, B. Schackel, ed., Elsevier-Science, 39-45

Carroll, J.M. (1990), 'Infinite Detail and Emulation in an Ontologically minimised HCI'. Proc. *CHI '90*, New York: ACM, 321-327

Carroll, J.M. and Campbell, R. (1989), 'Artifacts as Psychological Theories: the Case of Human-Computer Interaction'. *Behaviour and Information Technology*, **8**, ACM press, 247-256

Carroll, J M and Kellog, W A (1989), 'Artifact as Theory-nexus: Hermeneutics Meets Theory-based Design', *Proceedings CHI 89, Human Factors in Computer Systems*, ACM press, 7-14

Carroll, J.M. Koenemann-Belliveau, J. Rosson, M. and Singley, M. (1993), 'Critical Incidents and Critical Threads in Usability Evaluation', in: Proc. *People and Computers viii*, J. Alty et al (eds.), Cambridge University Press, 279-292

Carroll, J.M. and Mack, (1983), 'Actively Learning to use a Word Processor', in *Cognitive Aspects of Skilled Typing*, W.Cooper (ed), New York: Springer Verlag

Carroll, J.M. and Mack, (1984), 'Learning to use a Word Processor: By doing, by Thinking, and by knowing', in *Human Factors in Computing*, Thomas, J. and Schneider, M. (eds.), Norwood: Ablex

Carroll, J.M. Mack, R.L. and Kellogg, W.A. (1988), Interface Metaphors and User Interface Design. In: *Handbook of Human-Computer Interaction*, M. Helander (ed.) Elsevier North Holland, 67-81

Carroll, J.M. and Mazur, S.A. (1986), LisaLearning. IEEE Computer, 91(11), 35-49

Chin, C. (1984), 'Lisa Software' , *Infoworld*, Jan 16, p 27

Clark H.H. and Schaeffer E.F. (1987), 'Collaborating on Contributions to Conversation', Language and Cognitive Processes, **2**, 19-41

Clark H.H. and Schaeffer E.F. (1989), 'Contributing to Discourse', Cognitive Science, **13**, 259-294

Clark, L (1991), 'The Use of Scenarios by User Interface Designers', in *proc. People and Computers IV*, Diaper, D. and Hammond, N. (eds.), Cambridge University Press,

Cullen, J. Bryman, A. and Trimble, E. (1987), 'Knowledge Acquisition for Expert Systems' , SERC project

Cullen, J. and Bryman, A. (1988), 'The Knowledge Acquisition Bottleneck: Time for Reassessment?' in *Expert Systems*, **5**(3)

Defanti, T. Sandin, D. and Cruz-Neira, C. (1993), 'A Room with a View', IEE Spectrum, **30**, 10, 30-33

Dix, A. Finlay, J. Abowd, G. and Beale, R. (1993), 'Human-Computer Interaction'. Englewood Cliffs, New Jersey, Prentice-Hall

De Jong, G and Mooney, R (1986), 'Explanation-based Learning: An Alternative View'. Machine Learning, 1, 145-176

Dillon, A. Sweeney, M. and Maguire, M. (1993), 'A Survey of Usability Engineering Within the IT Industry - Current Practice and Needs', in: *Proc. People and Computers viii*, J. Alty et al (eds.), Cambridge University Press, 81-94

Diaper, D. (1990), 'Analysing Focused Interview Data with Task Analysis for Knowledge Descriptions (TAKD)', *Proceedings Interact 90*, Diaper et al (eds), Elsevier North-Holland, 277-282.

Dix A, Finlay J, Abowd G, and Beale J. (1992), 'Human-Computer Interaction', Eaglewood Cliffs, NJ: Prentice Hall

Douglas, S and Moran, T. (1983), 'Learning Text-Editing Semantics by Analogy', *Proc.CHI-83*, A Janda (ed), ACM press, 207-211

Draper, S. (1986), Display Managers as the Basis for Human-Machine Communication, in: *User Centred System Design New Perspectives on Human-Computer Interaction* , D. Norman & S. Draper ed Lawrence Erlbaum Associates, New Jersey

Dutt, A. Johnson, H. and Johnson, P. (1984), 'Evaluating Evaluation Methods', in *Proc. People and Computers IX*, Cockton, G. Draper, S. and Weir, G. Cambridge University Press, 109-124

Erickson T.D., (1990), 'Working With Interface Metaphors' in *The Art of Human Computer Interface Design*, B. Laurel (ed), Addison Wesley, Reading MA, 65-74

Ericsson, K.A. Simon, H.A. (1980),Verbal Reports as Data,, in. *Psychological Review*, **87**, (3), 215-251

Ericsson, K.A. Simon H.A. (1984), 'Protocol Analysis', MIT press

Fitts, P and Posner, M. (1967), 'Human Performance', Belmont: Brooks Cole

Foss, D. Rosson, M. and Smith, P. (1982), 'Reducing Manual Labour: An Experimental Analysis of Learning Aids for a Text Editor', in *Proc. Human Factors in Computing Systems Conference*, Gaithersberg, Maryland

Galambos, J. (1986), 'Knowledge Structures for Common Activities', in *Knowledge Structures*, Galambos, J. Abelson, R. and Black, J. (eds.) Laurence Erlbaum Associates

Gaver, W. (1991), Technology Affordances, *proc. CHI 91: Reaching Through Technology*, Robertson, S. Olson, M. and Olson, J (eds.) ACM Press, 79-84

Gentner, D. (1983), 'Structure Mapping: A Theoretical Framework for Analogy', Cognitive Science, **7**, 155-170

Gittens, D. (1986), 'Icon-based Human Computer Interaction'. *International Journal of Man-Machine Studies*, **24**, 519-543

Gould, J. D. and Lewis, C. (1985), 'Designing for Usability - Key Principles and What Designers Think'.*Communications of the ACM*, **28**, 300-311

Gray, W. John, B. Stuart, R. and Lawrence, D. (1990), 'GOMS Meets the Phone Company: Analytic Modeling Applied to Real-World Problems', *Proc. Interact 90*, Diaper et al (eds), Elsevier North-Holland, 29-34

Green, A. and Barnard, P. (1990), 'Iconic Interfacing: The Role of Icon Distinctiveness and Fixed or Variable Screen Locations', *Proc. Interact 90*, Diaper et al (eds), Elsevier North-Holland, 457-462

Grudin, J. (1989), 'The Case Against User Interface Consistency', *Communications of the ACM* , **32** (10)

Halasz, F. and Moran, T. (1982), 'Analogy Considered Harmful', in *Proc. CHI '82 Human Factors in Computing Systems*, New York, ACM Press, 383-6

Hartson, H. Siochi, A. and Hix, D. (1990), 'The UAN: A User-Orientated Notation for Direct Manipulation Interface Designs', *ACM Transactions on Information Systems*, **8**(3), 181-203

Hollan, J.D. Hutchins, E. James, D. and Norman, D.A. (1984), 'STEAMER: An interactive inspectable simulation-based training system', *AI magazine*, Summer 1984, 15-27

Hollnagel, E. (1993), 'Human Reliability Analysis: Context and Control', Academic Press

Howes, A. and Payne, S. (1990), 'Display-based Competence: Towards User Models for Menu-Driven Interfaces', *International Journal of Man Machine Studies*,

22, 365-395.

Hutchins, E.L. (1986) 'Metaphors for Interface Design'. Presented at Workshop on Multimodal Design, Venaco, Corsica, France

Hutchins, E.L. Hollan, J.D. Norman, D.A. (1986), 'Direct Manipulation Interfaces'. in: *User Centred System Design New Perspectives on Human-Computer Interaction*, D. Norman & S. Draper (eds.) Lawrence Erlbaum Associates, New Jersey, 31-62

Jacob, R. (1982), 'Using Formal Specifications in the Design of a Human-Computer Interface', *Proceedings of a Conference on Human Fcators in Computing Systems, Gaithersburg, Maryland*

Jeffries, R. Miller, J. Wharton, C. and Uyeda, K. (1991), 'User Interface Evaluation in the Real World: A Comparison of Four Techniques', *proc. CHI 91: Reaching Through Technology*, Robertson, S. Olson, M. and Olson, J (eds.) ACM Press, 119-124

Johnson, P. (1992), 'Human-Computer Interaction: Psychology, Task Analysis and Software Engineering'. London: McGraw-Hill

Johnson, H. and Johnson, P. (1991), 'Task Knowledge Structures: Psychological Basis and Integration into System Design', Acta Psychologica, 78, 3-26

Johnson, P. Johnson, H, Waddington, R. and Shouls, A. (1988), 'Task-related Knowledge Structures: Analysis, Modelling and Application', in: *People and Computers IV*, D.M. Jones and R. Winder (eds.), Cambridge University Press, 35-62

Johnson, P. Diaper, D. and Long, J. (1984), 'Tasks, Skills and Knowledge-Based Descriptions', in *INTERACT 84 - The First IFIP Conference on Human-Computer Interaction*, B. Schackel (ed.), Elsevier-Science

Johnson-Laird, P. (1983), 'Mental Models', Cambridge University Press

Johnson-Laird, P. (1988), 'The Computer and the Mind', Cambridge MA, Harvard University Press

Jones, M. (1990), 'Mac-Thusiasm: Social Aspects of Microcomputer Use', *Proceedings Interact 90*, Diaper et al (eds), Elsevier North-Holland, 193-198.

Karat, J. and Bennet, J. (1991), 'Using Scenarios in Design Meetings'. in *Taking Software Design Seriously* (Karat, J., ed), London: Academic Press

Karat, C-M. Campbell, R. and Fiegal, T. (1992), 'Comparison of Empirical Testing and Walkthrough Methods in User Interface Evaluation', in: *proc. CHI 92*, P.Bauersfield et al (eds.), 381-388

Kellogg, W (1990), 'Qualitative Artifact Analysis', *Proc. Interact 90*, Diaper et al (eds), Elsevier North-Holland, 193-198.

Kieras, D.E. (1988), 'Towards a Practical GOMS methodology for User Interface Design', In: *Handbook of Human-Computer Interaction*, M. Helander (ed.) Elsevier North Holland 135-156

Kieras, D.E. and Polson, P.A. (1985), 'An Approach to the Formal Analysis of User Complexity', *International Journal of Man Machine Studies*, **22**, 365-395.

Kilgour, A. (1989), 'Metaphorical Scaffolding for Interface Builders' Presented at BCS/HCI Workshop on Visual Languages, October

Knowles, C. (1988), 'Can Cognitive Complexity Theory (CCT) Produce an Adequate Measure of System Usability?', in: *People and Computers IV*, D.M. Jones and R. Winder (eds.), Cambridge University Press, 291-307

Lakoff, G and Johnson, M (1980), 'Metaphors we Live by', University of Chicago Press

Laurel, B. (Ed.), (1990), The Art of Human Computer Interface Design, Addison Wesley, Reading MA.

Lewis, C Polson, P Wharton, C and Reiman, J (1990), 'Testing a Walkthrough methodology for Theory-based Design of Walk-up-and-use Interfaces', *Proc. CHI-90*, J R Chew and J Whiteside (eds), ACM press, 235-241.

Lewis, C. (1986), 'A Model of Mental Model Construction'. *Proc. CHI '86 Human Factors in Computer Systems*, New York ACM, 306-313

Lewis, C. (1988), 'Why and How to Learn Why: Analysis-Based Generalisation of Procedures', Cognitive Science, **12**, 211-256

Macaulay, L. (1995), 'Human-Computer Interaction for Software Designers', International Thompson Computer Press

Mack, R.L. Lewis, C. and Carroll, J.M. (1984), 'Learning to use Word Processors: Problems and Prospects', *TOOIS*. New York: ACM Press

MacLean, A. Bellotti, V. and Young, R. (1990), What Rationale is There in Design, *Proceedings Interact 90*, Diaper et al (eds), Elsevier North-Holland, 207-212.

Maguire, M. and Sweeney, M. (1989), 'System monitoring: Garbage generator or Basis for comprehensive monitoring system', *People and Computers V*', A G Sutcliffe and L Macaulay
375-395, Cambridge University Press

Maissel, J. Dillon, A. Maguire, M. Rengger, M. and Sweeney, M. (1991), 'Context Guideline Handbook', National Physical Laboratory, Teddington, UK, MUSiC Project Deliverable IF2 2 2.

Mayes, J.T. Draper, S.W. MacGregor, A. and Oatley, K. (1988), 'Information Flow in a User Interface: The Effect of Experience and Context on the Recall of MacWrite screens', in: *People and Computers IV*, D.M. Jones and R. Winder (eds.), Cambridge University Press, pp 275-289

Mayo, K. and Hartson, R. (1993), 'Synthesis-Orientated Situational Analysis in User Interface Design' in *Springer-Verlag Lecture Notes in Computing*, **753**

Mitchell, T. Keller, R. and Kedar-Cabelli, S. (1986), 'Explanation-based Generalisation: A Unifying View', *Machine Learning*, **1**, 47-80

Monk (1986), 'Mode Errors: A User-Centred Analysis and Some Preventative Measures Using Keying Contingent Sound', in *International Journal of Man Machine Studies*, **24**

Monk, A. Wright, P. Haber, J. and Davenport, L (1991), 'Improving Your Human-Computer Interface', Prentice Hall

Moran, T P (1983), 'Getting Into a System: External-Internal Task Mapping Analysis' *Proceedings CHI-83*, A Janda (ed), ACM press, 45-49.

Mountford, J. (1990), 'Tools and Techniques for Creative Design' in *The Art of Human Computer Interface Design*, B. Laurel (ed) Addison Wesley, Reading MA, 17-30

Nelson, T. (1990), 'The Right Way to Think About Software Design', in *The Art of Human Computer Interface Design*, B. Laurel (ed) Addison Wesley, Reading MA, 235-243

Newell, A. and Simon, H. (1972), 'Human Problem Solving'. Eaglewood Cliffs N.J.: Prentice-Hall

Nielsen, J (1990), 'Traditional Dialogue Design Applied to Modern User Interfaces', *Communications of the ACM*, 33, 109 - 118.

Nielsen, J (1992), 'Finding Usability Problems Through Heuristic Evaluation', *Proceedings CHI-92*, P Bauersfeld et al (eds.), 373-380.

Nielsen, J and Molich R (1990), 'Heuristic Evaluation of User Interfaces', *Proceedings CHI-90*, J R Chew and J Whiteside (eds.) 249-256, ACM press.

Nielsen, J and Phillips, V (1993), 'Estimating the Relative Usability of Two Interfaces: Heuristic, Formal, and Empirical Methods Compared' *Proc. INTERCHI 93*, Ashlund, S. Mullet, K. Henderson, A. Hollnagel, E. and White,.T(eds.) ACM Press, 214-221

Norman, D.A. (1981), 'The Trouble With Unix', Datamation,27, 556-563

Norman, D.A. (1986), 'Cognitive Engineering', in: *User Centred System Design: New Perspectives on Human-Computer Interaction*, D Norman & S Draper (eds.) Lawrence Erlbaum Associates, New Jersey, 31-62

Norman, D.A. (1988), 'The Psychology of Everyday Things' New York: Basic Books

Ogden, W. and Boyle, J. (1982), 'Evaluating Human-Computer Dialogue Styles: Command vs. form/fill-in for Report Modification', *Proc. Human Factors Society, 26 annual meeting, Santa Monica, CA*, 542-545

O'Malley, C. and Draper, S. (1989), 'Representation and Interaction: Are Mental Models All In The Mind', in *Models in the Mind, Theory, Perspectives and*

Application, Rogers, Y. Rutherford, A. and Bibby, P. (eds.), London: Academic Press, 73-92

Olphert, C.W. (1986), 'Case Studies of Usability', in 'Designing Usable IT Products' HUFIT project, ETW '86 seminar, Loughborough UK, 1/10/1986

Pakin, S. and Wray, P. (1982), 'Designing Screens for People to use Easily', *Data Management*, July 1982, 36-41

Paap, K.R. and Roske-Hofstrand, R.J. (1988), 'Design of Menus' In: *Handbook of Human-Computer Interaction*, M. Helander (ed.) Elsevier North Holland, 205-233

Payne, S.J. (1984), 'Task-action Grammers', in *INTERACT '84 - First IFIP Conference on Human-Computer Interaction*, Shackel, B. (ed), Elsevier Science, Amsterdam

Payne, S.J. (1987), 'Complex Problem Spaces: Modelling the Knowledge Needed to Use Interactive Systems', in *Proc. Interact 87*, Amsterdam: Elsevier Science Publishers

Payne, S J (1990), 'Looking HCI in the I', *Proceedings Interact 90*, D.Diaper et al (eds), Elsevier North-Holland, 185-192.

Payne, S J (1991), 'Interface Problems and Interface Resources', in *Designing Interaction*, J M Carroll (ed), Cambridge University Press.

Payne, S. and Green, T.R.G. (1986), 'Task action grammars: A Model of the Mental Representation of Task Languages', *Human Computer Interaction*, 2, 93-133

Pazzani, M. (1987), 'Inducing Causal and Social Theories: A Prerequisite for Explanation-based Learning', *Proc. Fourth International Machine Learning Workshop*, Irvine CA, 230-241

Polson, P. and Lewis, C. (1990), Theory based design for easily learned interfaces. In: Human-Computer Interaction **5**, 191-220

Polson, P Lewis, C Reiman, J and Wharton, C (1992), "Cognitive Walkthroughs: a Method for Theory-based Evaluation of User Interfaces", *International Journal of Man Machine Studies*, Vol 36, pp 365-395.

Preece, J. Sharp, H. Benyon, D. Holland,S. and Carey, T. (1994), 'Human-Computer Interaction', Addison Wesley

Rasmussen, J. (1986), 'On Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering', Amsterdam: Elsevier

Ravden S. and Johnson G. (1989), 'Evaluating Usability of Human Computer Interfaces', Ellis Horwood

Reason J. (1986), 'Framework Models of Human Performance and Error: A consumer guide', *Mental Models, tasks and Errors*, I Goodstein H B Andersen and S E Olsen (eds.), Taylor and Francis, London

Reisner, P. (1981), 'Formal Grammers and Human Factors Design of an Interactive Graphics System', IEEE Trans. on Software Engineering, SE-7 (2), 229-240

Reisner, P. (1982), 'Analytical Tools for Human Factors of Software', in *End User Systems and Their Human Factors*, Blaser, A. and Zoeppritz, M. (eds), Springer-Verlag, Berlin, 94-121

Reisner, P. (1990), 'What is Inconsistency?', *Proceedings Interact 90*, D.Diaper et al (eds), Elsevier North-Holland, 175-181.

Rogers, Y. (1989), 'Icon Design for the User Interface', *International Review of Ergonomics*, 129-154

Rosch, E. (1975), 'Cognitive Representations of Semantic Categories', *Journal of Experimental Psychology: General*, **104** (3), 192-233.

Rosch, E. Mervis, C Gray, w. Johnson, D. and Boyes-Braem, P. (1976), 'Basic Objects in Natural Categories', *Cognitive psychology*, **8**, 382-439

Roth, I. and Frisby, J. (1986), 'Perception and Representation', Open University Press

Rowley, D. (1994), 'Usability Testing in the Field: Bringing the Laboratory to the User', in *Proc. CHI 94*, Plaisant, C (ed.), ACM Press

Rubenstein R. and Hersch, H. (1984), 'The Human Factor: Designing Computer Systems for People', Burlington MA, Digital Press

Rutkowski, C. (1982), An Introduction to the Human Applications Standard Computer Interface, Part 1: Theories and Principles, *Byte* , 7 (11), 291-310

Schank, R. (1982), 'Dynamic Memory: A Theory of Reminding and Learning in Computers and People', Cambridge University Press

B Shackel (1986), 'Ergonomics in design for usability', *People and Computers: Designing for usability*, M D Harrison and A F Monk (eds.) Cambridge University Press

Shackel, B. (1991), 'Usability - Concept, Framework, Definition Design And Evaluation', in *Human Factors for Informatics Usability*, Shackel, B. and Richardson, S. (eds.), Cambridge University Press, 21-37

Shneiderman, B. (1982), 'The Future of Interactive Technology and the Emergence of Direct Manipulation', *Behaviour and Information Technology*, 237-256

Shneiderman, B. (1983), Direct Manipulation: A Step Beyond Programming Languages, *IEEE Computer*, **16**(8), 57-69

Shneiderman, B. (1986), Designing the User Interface: Strategies for Effective Human Computer Communication, Addison-Wesley

Simon, T. (1988), 'Analysing the Scope of Cognitive Models in Human-Computer Interaction: A Trade-off Approach', in: *People and Computers IV*, D.M. Jones and R. Winder (eds.), Cambridge University Press, pp 79-93

Smith, D. Irby, C. Kimball, R. and Harslam, E. (1982), 'Designing the STAR User Interface', *Byte*, 7(4), 242-82

Smith, R, (1987), 'Experiences with the Alternate Reality Kit-an Example of the Tension Between Literalism and Magic', in *proc. CHI+GI' 87 Conference on Human Factors in Computing Systems*, Toronto, Canada, New York: ACM

Smith S L and Mosier J (1986), 'Guidelines for Designing User Interface Software', *Mitre Corporation Report MTR-10096*, Bedford, MA.

Stevenson, R.J. Manktelow, K.I. and Howard M.J. (1988), 'Knowledge Elicitation: Dissociating Conscious Reflections from Automatic Processes', in: *People and*

Computers IV, D.M. Jones and R. Winder (eds.), Cambridge University Press, 565-579,

Suchman, L. (1987), 'Plans and Situated Actions', Cambridge University Press.

Sutcliffe A.G. (1988), 'Human Computer Interface Design', Macmillan, London

Sutcliffe, A G and Springett, M V (1992), 'From User's Problems to Design Errors: Linking Evaluation to Improved Design Practice', *Proc. People and Computers vii,* A Monk et al (eds), Cambridge University Press, 117-131.

Tauber, M. J. (1986), 'Top-Down Design of Human-Computer Interfaces', *Visual Languages* (Chang et al eds.) New York: Plenum Press

Tauber, M. J. (1990), 'E-TAG: Extended Task-Action Grammer. A Language for the Description of the User's Task Language', *Proc. Interact 90*, D.Diaper et al (eds), Elsevier North-Holland, 163-168.

Te'eni, D. (1990), 'Direct Manipulation as a Source of Cognitive Feedback: a Human-Computer Experiment with a Judgement Task', *International Journal of Man Machine Studies*, **33**, 453-466.

Turkle, S. (1984), 'The Second Self: Computers and the Human Spirit', London: Granada

Virzi, R. (1990), 'Streamlining the Design Process: Running Fewer Subjects'. *Proc. 34th annual meeting of the Human Factors Society*, 291-294

Virzi, R. (1992), 'Refining the Test Phase of Usability Evaluation: How Many Subjects are Enough?', *Human Factors*, **34**, 457-468

Weiser, M. (1991), 'The Computer for the 21st Century', *Scientific American*, September, 66-77

Wharton, C. Bradford, J. Jeffries, R. and Franzke M. (1992), 'Applying Cognitive Walkthroughs to More Complex Interfaces: Experiences, Issues, and Recommendations', in *proc. CHI 92*, P.Bauersfield et al (eds.), pp 381-388

Whiteside, J. Bennett, J. and Holzblatt, K. (1988), 'Usability Engineering, our experience and evolutions'. in *Handbook of Human Computer Interaction*, M

Helander (ed.) Elsevier North Holland, 791-817

Williams, M. Hollan, J. and Stevens, A. (1983), 'Human Reasoning About a Simple Physical System', in *Mental Models*, Gentner, D. and Stevens, A. (eds.), Hillside: Erlbaum

Wilson, J. and Rutherford, A. (1989), 'Mental Models: Theory and Application', *Human Factors*, **31**, 617-634

Winston, P. (1982), 'Learning New Principles from Precedents and Exercises', *Artificial Intelligence*, **19**, 321-350

Woods, W. (1970), 'Transition Network Grammars for Natural Language Analysis', *Communications of the ACM*, **13**, 591-606

Wright, P and Monk, A F (1989), 'Evaluation for Design', *People and Computers V.*, A G Sutcliffe and L Macaulay (eds); Cambridge University Press, 345-358.

Wright, P. Monk, A and Carey, T. (1989), 'Cooperative Evaluation - The York Manual Version 0.4.' Dept of Psychology, University of York, UK

Wright, P and Monk, A F (1991), 'A Cost-effective Evaluation Method for use by Designers', *International Journal of Man Machine Studies*, **35** (6), 819-912.

Young, R and Barnard, P. (1987), 'The use of Scenarios in Human-Computer Interaction Research: Turbocharging the Tortoise of Cumulative Science', in *proc. CHI+GI '87: Human Factors in Computing Systems and Graphics Interfaces*, Carroll, J. and Tanner, P. eds. ACM Press 291-296.

Young, R. Howes, A. and Whittington, J. (1990), 'A Knowledge Analysis of Interactivity', *Proceedings Interact 90*, D.Diaper et al (eds), Elsevier North-Holland, 115-120

Yourdon, E (1989), 'Structural Walkthroughs', 4th edition, Englewood Cliffs, NJ: Yourdon Press.

Bibliography

M.V. Springett (1992), 'The Utility of User Action Models For Direct Manipulation Design' Proc. *IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction*, C. Unger and J.A. Larsen eds., Elsevier North-Holland.

A.G. Sutcliffe and M.V. Springett (1992), 'From User's Problems to Design Errors: Linking Evaluation to Improving Design Practice' *People and Computers vii*, A. Monk, D. Diaper and M.D. Harrison eds., Cambridge University Press.

M. Springett and A.S. Grant: (1992), 'Interface Semantics and Users' Device Models: Identifying Evaluation Issues for Direct Manipulation Design'. Proc. *People and Computers viii*, J. Alty et al (eds.), Cambridge University Press, pp 249-266, [1993].

Appendix A

The MMA Method as Presented to Subjects

Evaluation Method

Introductory Booklet

Introduction

This method helps you to perform usability evaluation of interfaces in collaboration with an experimental user subject in a mock task-session. The method is a simple process of analysing user problems, and suggesting design change in the light of that analysis. It involves recording the problems that a user has with a particular action, and comparing what they wanted or expected with what the system actually provides. Here is a broad outline of the process, which is fully described in the method manual booklet.

• Before the Session
You will be given a set of instructions and an experimental task. Read the instructions to the user, and present the task.

• During the Session
The user will perform the task, verbalising continuously. You record any problems that the user encounters, asking questions if necessary.

• With the user After the Session
You clarify the user's view of each problem incident (the user may re-examine the screen at this point, if desired)

• After the User Leaves
i. You check the user's view of each incident against the actual design and presentation of it, to pinpoint the cause of the problem.

ii. You recommend ways of solving the design problems implied by the problem incidents.

A further diagnostic aid is available for use by more expert evaluators, although it is not included in this Introdutory Booklet. This consists of a model of interaction which goes into greater detail of user behaviour than the model presented to first time evaluators in this booklet. The more detailed model provides an aid to deeper diagnosis, accounting for differing levels of user expertise, and types of knowlege recruited by the user. It is recommended that evaluators with a background in Evaluation or Human-Computer Interaction refer to the more advanced model.

1. Understanding the Method For Interpreting User Problems

The method provides criteria for interpreting the problems that the user encounters when using the system. This is done by providing a basic description of the type of activities that the user engages in within the course of a single action, the information needs that the system must supply to support those activities, and ways that this can fail. Reading this description will assist you in interpreting problem incidents that the user reports during the session.

The four key elements of the session are:

1. What the user does, both mentally and physically during performance of each action on the machine (activities)

2. What the system must provide to support this (roles)

3. How this can go wrong (mismatches)

4. How elements of design may be changed in order to remove the problem (solutions)

Activities

There are four basic phases of activity within each user action. These are:

1. Deciding what one wants to do (the task-goal)

There is an important distinction between a 'task-goal' and a 'device-goal'. The task-goal is what one wishes to achieve for ones own purposes, such as sending mail to a friend. The device goal is the specific action(s) that it takes (such as a 'send' command on an e-mail system) to achieve a task-goal. It is likely that a single task-goal will include more than a single action. For example, sending mail may involve dragging a menu, selecting a command and typing the name of a file.

2. Deciding how it can be done

This is the phase of translating the task-goal to a device-goal or goals. The user must decide if the task goal is possible, and what must be done to achieve it using the system's facilities. The user may feel that previous actions or knowledge of the overall system can tell them all they need to know. Also, the user's current task-goal may include a sequence of actions that the user believes will go together. However, especially for new or occasional users, the interface will need to provide specific hints about what to do, and the features to use to do it. On the type of system we are looking at, this is likely to be textual or pictorial cues representing features, or the 'familiar' appearance of , for example, a desktop on the screen.

3. Executing the action

The user performs a keyboard or mouse action. It is likely that some degree of skill is required, particularly for the mouse actions, although these should be skills that are easily within user capabilities.

4. Assessing the Result

This phase involves assessing the new state caused by the action. The user wants to compare the change to what was desired. Also, the user will need to see if the next or subsequent actions are possible. The visible result of the action is referred to as the 'feedback'.

Figure 1 shows a flow-diagram of the activity sequence within an action. Within the four activity phase are seven specific activity types. The seven individual activities correspond to interface 'roles' described below.

Roles and Mismatches

This section describes the roles that the interface must play in providing the user with sufficient information to proceed with action and satisfy goals. Each section also has a description of the range of ways in which mismatches can occur.

Why roles?
Roles are basic types of support that the interface must provide for each action. The roles correspond to component user activities within each action. The term roles is used because it is a neutral term which recognises the variety of ways in which information and knowledge requirements can be satisfied. For example, if the user is using a new feature, but knows where features of that type are situated on the screen, the role of locating a feature (described fully below) is fulfilled. If this type of information is not known, the user relies on specific items such as dedicated screen areas, menu headers, names and iconic symbols to help locate and identify features.

Why Mismatches?
Interaction on a display-based system may be seen as dialogue, similar to a conversational dialogue in that its success depends on the two participants establishing a mutual understanding. For example, if a participant in a conversation says 'pick up the third block from the left', that may mean from the left as the speaker looks or (from the point of view of someone on the other side of the blocks) the third one from the right. The hearer may pick up the correct block (a match) or there may be confusion over *whose* left the speaker refers to (a mismatch). This is a useful metaphor for the user/system dialogue. The continuous cycle of user search, interface signals, user action, interface response, and further user action is prone to mismatches. For example, the interface, by referring to a facility with the wrong name or icon, may cause the user to choose the wrong action. Similarly, by producing a misleading image as feedback to an action, the system may cause a mismatch by confusing them about whether or not a goal has been satisfied. The problem of mismatches between system and user is investigated using this evaluation method.

Detailed Descriptions of Roles and Mismatches

What follows is a description of each role that *some aspect* of the interface must fulfil in order to support user action. They are linked to types of mismatch that can cause role failures. Figure 2 shows types of mismatch associated with each role. These mismatch types are described more fully in the following section.

Role for Supporting Goal Formation

The **Goal Formation** role refers simply to the provision of a needed function for a particular action. The system must support the task goals that the user expects and wants. For example, a text editor without a cut and paste facility would probably be seen as incomplete by most users, even though it could function without one.

Roles for Deciding How One Can Achieve a Goal

This phase has three component roles. These are **locate feature, identify feature**

and **Specify action.**

Locate Feature
The **locate feature** role becomes important where the user has a notion of what is needed but it is not aware precisely of the required feature to use. The locate feature role is the interface's role in constraining this search by indicating where the user should look. For example, a menu header which can be seen by the user and matched to the type of item that is is being looked for, would serve as good locating information.

Mismatches associated with the Locate Feature role are:

• Insufficient locating information:
The user is searching for a feature in the wrong place, or does not know where to search

• Misleading locating information
The user reasons that the feature is in a certain place, because the interface is giving a misleading impression

Identify Feature
Along with locators to help search, the user will require feature identifiers. These are typically names of menu items or icon arrays. The success of the **identify feature** role depends on the user's ability to match what they express with what is required. The user will have a mental picture (however vague) of what is required. The role of the feature identifier is to effect recognition of the appropriate feature by matching, using either text or an image, characteristics of what the user is looking for.

Mismatches associated with the Identify Feature role are:

• Lack of meaning
The user passes over a needed or useful feature because the cue does not alert the user to the feature's utility

• Confusion between features
A feature's cue appears relevant to the current task-goal when it is not

Specify Action
The **specify action** role is the most complex role. This is the initial phase of thinking about the goal, and an individual action within it, as an action on the system. In a sense this involves a translation from a task (e.g. drawing or writing) that the user may know, into the specifics of how it is done on the system. It is complex because potential mismatches may be the result of previous actions or user interpretation of high-level principles, and therefore be distanced from the overt problem that the user encounters. The user may reason from previous interaction, general knowledge of the task, assumptions about general system principles or knowledge of other systems.

Mismatches associated with the Specify Action role

There are three categories of mismatch here. These are:

• Failure to support the user's view of the task:
These can be observed when the user expresses frustration or confusion due to a

failure to comprehend, for example, the default settings or conventions. To take a hypothetical case, a user of a draw package finds that they cannot plot a point with precision. Unknown to them is that the default state of the package is an invisible plotting restriction (or 'grid'). This principle is missed by the user because it is unexpected and counter to the 'drawing' metaphor. In this example the default state embodies a notion of the 'drawing-on-paper' metaphor which does not match with the user's notion. Similarly, a principle of operation, such as the procedure for selecting colours or sizes may force errors if it is incomprehensible or counter to what the user would naturally expect.

• Hidden effects of prior action:
These are the result of feedback perception mismatches from a previous action. If, for example, the user has set a default without realising, in the course of using a feature, the feature may behave differently next time, without the user realising why.

• Unsupported assumptions from prior action:
The user may express expectations from previous actions that led to an erroneous action. For example, if a user has selected a type of line drawing facility from a menu, and it cannot be operated on in the same way as a previously selected option, the user may be confused or regard this as inconsistent.

Role for Executing

The **Execution** role consists of two principles. The first is that the response to an input should seem 'natural' to the user, for example, cursor movements corresponding to the user's hand movements with the mouse. The second is that the action should not place unreasonable demands on, for example steadiness of hand or eye-tracking. Some actions are, by their nature, difficult. For example, drawing a straight line is hard (hence the need for rulers), and seeing precisely where the boundaries of an item are places demands on perceptual capacities. The system can reasonably be expected to provide facilities to ease the difficulty (e.g. constrained line drawing).

Mismatches associated with the Execution Role:

• Lack of support for difficult action
The user expresses difficulty in performing with a feature to a sufficient level of satisfaction

• Unnatural input response
The user expresses that the behaviour of a feature is unexpectedly awkward

Roles for Assessing the Result of the Action

This is split into two roles, **Perceive feedback** and **Comprehend feedback.** The interface must provide a perceptible indication of a state change, and this must be comprehensible in terms of the user's goal.

Mismatches associated with the Perceive Feedback role:

• Absent/Delayed Feedback

Changes that are not indicated with feedback are likely not to be known by the user, who will subsequently specify action with an incorrect model of some aspect of the system-state. Equally, a delay may cause confusion, as the user will immediately monitor for changes and proceed to the next action.

• Brief/Obscure feedback

The designers may provide feedback which is too small, brief or indistinguishable to alert the user.

Mismatches associated with Understand Feedback role:

• Failure to reflect changes in system-state
(e.g. secondary effects)

The system may fail to give an accurate visual indication of a state change. For example, a delete action may seem not to have worked because an image of the deleted item (or a remnant of it) remains on the screen.

• Failure to reference user goals

The user will relate the result of an action to the intended task-goal. If the change does not clearly confirm progress, the user may become confused. An example would be a 'cut' facility without a facility for seeing precisely what has been cut (such as a 'clipboard' or buffer). The user would probably see that there has been a cut, but could have trouble confirming that the intended items were cut and that they still exist in a buffer.

2. Identification of Problems

You will be presented with a form describing the way in which role mismatches can be identified during and after a session. The procedural instructions will describe the process of identification in explicit detail. What follows is an outline description of how types of incident and user reporting identify types of role mismatch. You will be able to gain the data first by quizzing the user during or after the session. Examples of each type of role mismatch in a mock-up Incident Record Sheet are presented with the descriptions. The examples are taken from a real evaluation of a drawing package. The items in brackets are notes referring to actual procedures and states, contrast with the users' model. You do not need to write down this information on the Incident Record Sheets if you feel this is unnecessary. What is important is to capture the essence of the problem.

Goal Formation Mismatches

Simply identified by the user declaring a task-goal that is not, in fact, supported by the interface.

Description of Incident							
	user tries to rotate a text item (can't be done)						
Repeats of same problem	/	/					
List any alterations/corrections?	**realises it can't be done**						

In the example displayed above the user is clearly trying an impossible action. Therefore, in the absence of any reason to the contrary, the evaluator *recommends* functionality to support such an action. The user repeated the error twice more (recorded in the boxes provided), and reported after the session that he had finally accepted it could not be done.

Locate Feature Mismatches

The user searches in the wrong place for a feature, perhaps naming an area of the interface.

The user declares difficulty in deciding where to search, or disappointment with failed search.

Description of Incident							
	can't find clipboard display after cutting an item **looking in wrong menu**						
Repeats of same problem							
List any alterations/corrections?	**found eventually in another menu**						

In this example, the user searched and failed to find the clipboard, not because the icon was poor (as the eventual discovery suggests), but because the wrong menu was assumed as the obvious location to look in. Therefore the evaluator *specifies* moving the option to the same menu as the cut and paste options with which it is connected.

Identify Feature Mismatches

The user uses an irrelevant or suboptimal feature (or verbally misidentifies a feature).

The user finds but does not recognise a needed feature from its cue.

Description of Incident
wanted an arrow pointing left on a drawn line, picks menu option with arrow facing left, arrow appears facing right
Repeats of same problem ⁄ ⁄ ⁄ ⁄ ⁄ ⁄ ⁄
List any alterations/corrections?

In this example the user is repeatedly trusting. a cue which depicts a left-pointing arrow (an image of the goal-state). So the evaluator *recommends* that cue and function be made to resemble each other (the option of altering feature or function is left open at this stage, and the evaluator is therefore recommending rather than specifying a solution).

Specify Action Mismatches

Usually identified by the user attempting an action which cannot be achieved, or expressing confusion at the result of an action. The user may state a belief about how a goal can be satisfied which has too few actions in it, or show ignorance of a *relevant principle*, or the existence of *supporting functionality* (it is up to you to decide whether a feature is not discovered because the locating and identifying information is poor, or because the task is not designed in a way that is comprehensible to the user). References to similarities with previous actions also indicate a problem. Expressions of confusion over system behaviour also suggest this type of mismatch.

Description of Incident
trouble plotting a point with the mouse for drawing
Repeats of same problem
List any alterations/corrections?

In this example the user is unable to plot a point with the required precision, and fails to work out why this should be. In fact, the default state of the machine activates a plotting restriction which has to be removed by selecting a menu option. The problem is that the user has no idea that this is the case, expecting a 'draw' package to allow pen and paper style drawing. There is no reason to think that any user's notion of pen and paper drawing tasks would include a plotting restriction. The evaluator therefore *specifies* that the default state be changed with the restriction becoming a selectable option.

Description of Incident
tries to put arrow on arc line using menu option. claims 'it worked before' having done it for straight lines (can only be done by constructing a line from arc & straight lines, adding an arrow)

Repeats of same problem	/	/	/				

List any alterations/corrections?

This example shows a user who knows one operation (drawing an arrowed line), extending it to another (an arc line), reasoning from the similarity of the two objects (the lines). The specified action is nested in a sequence that the user believes is isomorphic with a known sequence. The 'specify action' mismatch is demonstrated when the user's expectations are compared to the more complicated actual procedure where two lines need to be drawn. As with the previous example, the user's notion of how to do it and the system's notion are critically different. The evaluator *specifies* that the arrow option be made legal for arc as well as straight lines.

Execution Mismatches

User expresses difficulty with the actual manipulations.

User indicates that a facility is behaving strangely or counterintuitively during performance of the action.

Description of Incident							
	chose 'select all' followed by dragging of whole drawing, an intended stretch causes a contraction, has to start again						
Repeats of same problem							
List any alterations/corrections?	user was trying to drag the whole diagram down						

The user reports that a downward drag produced an upward cursor drag on the screen, ruining the user's composition. The execution mismatch is the failure of the cursor to move in the way that the mouse moved. The evaluator *recommends* the redesign of this feature to a more natural response to input.

Perceive Feedback Mismatches

These problems may only emerge in the action specification stage of a subsequent action. The user may miss feedback information, and make erroneous assumptions which affect further actions. Also, the user may indicate may that there is a lack of information about the effects of the action.

Description of Incident							
	draws wrong shape, blames selecting palette for deselecting (fails to complete action)						
Repeats of same problem	—	—	—				
List any alterations/corrections?							

The user continually makes the error of failing to confirm a selection. By practising this selection action after the session the evaluator notices that the difference between an initial select and a confirmed select is hard to spot, and is probably not seen by the user who carries on thinking the selecting action is complete. The *recommended* solution is for clearer feedback to distinguish initial and confirmed palette option selection.

Understand Feedback Mismatches

These are often cases of the user mistakenly believing that there is a problem as a result of an action where, in fact, there is not.

The user expresses incorrect assumptions drawn from the feedback.

The user expresses confusion at the apparent result of an action, and is unable to confirm progress.

Description of Incident
draws wrong shape, blames the selecting palette for deselecting **(fails to complete selecting action)**

Repeats of same problem							

List any alterations/corrections?

The user has not actually made an error here, but believes an error has been made. The appearance on the screen is of an item being removed, whereas the attempted action (placing a circle around an item) was successful. The evaluator therefore *specifies* that the screen image of 'filled-in' shapes should be altered to reflect the true nature of the system state.

3. Recommending Solutions

There are four stages in the analysis of collected data. You will be asked to get as far through these stages as you feel able to. The stages are:

- Identifying a problem (done during the session)
- Pinpointing the cause (investigated after the session)
- Recommending a Solution (stating in general terms the change that should be made)

or

- Specifying a Solution (giving specific advice, such as a new facility, default state or menu name).

You may or may not feel able to specify a solution for all problems, but you are encouraged to do so if you feel that you can. You will be asked to briefly explain the reasoning behind your solution suggestions.

Examples of Completed Solution Sheets

Below are two examples of completed sections of a Change Suggestion Sheet. The first illustrates a *specified* solution

Solution:	Default state should not include an automatic plotting restriction. Restriction as option only.
Reasons:	current default confuses the user and makes plotting awkward

This is the example from the first 'Specify Action Mismatch' above. The evaluator is clear about the precise change that should be made. A specific alteration is referred to. A brief reference to the reason is also made, namely that users will have no idea why they cannot plot points precisely.

The second example is of a less specific *recommended* solution.

Solution:	Bring cues for arrows into line with the actual function of the options
Reasons:	Current design causes users to continually select the wrong option

This is the 'Identify Feature Mismatch' example. The evaluator leaves open the possibility of altering the feature's behaviour, the cue, or both. Again, a brief note referring to the reasoning behind the solution suggestion is entered below the solution.

Method Manual

1. Preparing for the session

Please check that you have copies of the following documents

• The Instruction Sheet for the user

• Incident Record Sheets (to record incidents during the session, and clarify after the session)

• Figure 2 from the 'Method Theory' booklet (role/mismatch reference diagram)

• Design Alteration Sheets

You are also provided with a copy of the 'Advice on Recording and Analysing' section from the 'Method Theory' booklet to use for reference.

2. When the User Arrives

1. Read the Instruction Sheet to them.

2. Present them with the task (the task sheet will be supplied).

3. Commence the session.

3. During the Session

• Observe the user, and listen for indications of problems

• Intervene with clarification questions if you need a clearer explanation

• Record problem descriptions on the Incident Record Sheet

With the User After the Session

• Conduct a clarification interview for <u>each recorded incident,</u> prompting the user by reminding them what they said at the time of the incident. Record corrections (if appropriate) or any alterations to the user's beliefs about how to do it, in the space marked 'any alterations/corrections?'. Duplicate errors can be recorded by ticking the 'repeats of same problem' boxes.

Ensure That You Are Clear About:

1. What they wanted to do

2. How they believed it would be achieved

3. What they found to be unsatisfactory/unexpected

4. Whether their view of how to do it has now changed

After the User Has Left

For each incident recorded:

• Compare the user's view of how to achieve the goal with the actual procedure for achieving it (a manual is provided if you are unsure).

• Examine the cause of the problem in terms of role mismatches.

• Enter your diagnosis and recommendations/solutions on copies of the Design Alteration Sheet (1 incident per sheet).

Filling in the Design Alteration Sheet

Please include the following when filling in Design Alteration Sheets:

• The incident number (from the order in which they happened)

• Your specification/recommendation for improved design (if you feel that a finding should lead to a change that would effect other actions and goals, such as the altering of a default principle, please state this)

• A brief description of your reasons for the suggestion (one or two sentences at most)

Instructions to the User

1. When the user enters

Present the user with the User Description Sheet

2. When the user Description Sheet is Completed

Say –You will shortly be presented with a task-sheet. The task-sheet contains an end-state of a text-editing task which you are asked to try to achieve. The file that you will be editing has the same document in a different state. By checking the task-sheet you will see the changes that you are asked to make. You are asked to perform this using the package. Try to perform the task to the best of your ability, as if it were a real task in a working environment. The session does have a time limit, but do not hurry. Perform at what you would consider to be a normal speed for such a task. The package is a Word Processing package, which also allows you to construct non-text items such as diagrams and add them to the text.

Please describe what you are doing as you perform the task. In particular, describe what you are intending to do next, and mention if you are having difficulties, or are dissatisfied with something.

Please make it clear whether any dissatisfaction or difficulties that you may have are with:

- Trying to decide what to do next
- Actually performing an action
- The result of an action that you have performed.

You may be asked to clarify some points when you report difficulties or dissatisfaction. You may, for example be asked to describe the reasons behind your decision-making. This does not imply that your reasoning or decision-making is wrong, and it should not be allowed to influence your behaviour in the session (it is important that your decision making is free from any such 'unnatural' influence).

Please turn the task-sheet over.

You are asked to double-click on the icon for the file 'TASK' which is located near the bottom of the screen. Enter the 'file' menu, and select the 'save as' option. Alter the filename to your initial and surname. Then commence the task.

Appendix B

Materials given to Subjects using UC Method

Introduction to the Evaluation Method

The Usability Checklist Evaluation Method

The accompanying documents describe the Checklist Evaluation Method. The contents of the document are as follows:

- A description of your role in the evaluation session
- A description of your (user)evaluator's role

In the accompanying document package:

- Overview of the Method (Marked 2)
- Instructions for completing the checklist (marked 3)
- The checklist
- Full descriptions of each checklist question (marked 4)
- Observer Instructions for the Session
- The procedure for recording design suggestions

1. Your Role

In essence, your role is to supervise and monitor an evaluation session. You will be monitoring the interaction behaviour of an 'evaluator' (a user) who will be trying example tasks on Microsoft Word for the Apple Macintosh. You will be asked to brief the evaluator before the session on what is required (the evaluators role is described below). You will then watch the session, taking any observational notes that you feel may be useful. Please do not communicate with the evaluator whilst the session is taking place.

After the session you will present the evaluator with the evaluation checklist. You are provided with detailed definitions of each checklist question, and should inform the evaluator that you will to read definitions aloud on request. You may wish to monitor the forms as they are completed to check for readability. You are, however, asked not to 'cross-examine' the evaluator.

The evaluator will leave after the checklist has been completed. You will then be asked to examine the answers provided, and make design change suggestions on the sheets provided. You will be asked to explain your motivation for the changes by citing the source of the information that led you to the decision.

2. The role of the 'evaluator'

Your evaluator will know nothing about the package which is to be used. The evaluator may declare other graphics, word-processor or computer experience before commencing the task. You will present the evaluator with a sheet detailing the task that

they will be performing. The evaluator will then attempt to perform the tasks, making notes when points of interest emerge. You will present the evaluator with the Checklist at the end of the session. The evaluator will complete the Checklist, requesting definitions of the Checklist questions if required.

Observer Instructions for the Session

This is the procedure for conducting the evaluation session.

1. Read the 'Before the Session' section of the 'Instructions to the User' sheet to the user

2. Present the task sheet to the user

3. Show user to the 'task' icon in the main directory

4. Observe the session, noting points of interest

5. When the session ends, read the 'After Session Instructions', and present the checklist. Read the 'Instructions for Completing the Checklist' section (marked 3) of the materials provided.

6. Provide definitions of checklist questions on request

(user leaves)

7. Use the completed checklist and your own observations to make design change suggestions, using the 'Design Alteration Sheets' (indicating the method's degree of influence on each decision).

Instructions to the User

Prior to the Session

(to be read to the user)

You will shortly be presented with a task. You are asked to perform this task using the package. Try to perform the task to the best of your ability, as if it were a real task in a working environment. The session does have a time limit, but do not hurry. Perform at what you would consider to be a normal speed for such a task. The package is a Word Processing package, which also allows you to construct non-text items such as diagrams and add them to the text.

You will be provided with a pen and paper. Please note anything that you feel would be of interest in an evaluation of the package's usability during the session.

You will be given a questionnaire at the end of the session. This will ask you about various aspects of the system you have used.

The first task is to go to the 'task' icon, and enter the file with a double click. Alter the name of the file to your initial and surname using the 'Save As' option from the 'File' menu. The task sheet shows the end state that you want to achieve. The file that you will open contains a document which you will need to edit in order to reach the end state.

INSTRUCTIONS TO READ TO THE USER AFTER THE TASK SESSION IS COMPLETE

Please fill in the checklist that is provided. Refer to your notes, or look again at the package that you have just used, where you feel the need to. Please ask for definitions of individual checklist questions if you are unsure. Write comments next to your checklist answers if you feel that extra explanation of your answers would be of use, or you feel particularly strongly about something.

(Read Section 3 'Instructions for Completing the Checklist' to the user)