CONSISTENCY AND USABILITY

Patrick William Jordan

Submitted for the degree of Doctor of Philosophy (Ph.D.) to the University of Glasgow Faculty of Science.

Research carried out in the Department of Psychology.

Submitted April 1993.

Dedicated to my parents: Bill and Jill

ABSTRACT

The thesis is divided into two main parts. In the first a multi-component framework for usability is outlined. The components — guessability, learnability, experienced user performance, system potential, and re-usability — describe how users' performance on a task changes with experience, and each is associated with a different part of a learning curve. Definitions of the components are offered, as are examples of where each may be particularly important. The potential advantages of the framework are then discussed. An experimental methodology and criteria based analysis method for quantifying the components of usability is proposed. This is demonstrated in the evaluation of a video cassette recorder. Non-experimental alternatives to this methodology are also considered.

The second part of the thesis is about the issue of consistency. A distinction between two types of consistency — set compatibility and rule compatibility — is proposed. Set compatibility is concerned with inter-task associations, whilst rule compatibility concerns the prior association of tasks and action-rules. Predictions are made about the effects of each type of consistency on the various components of usability, and these are tested in the context of a study of the invocation of menu commands. Results indicated that rule compatibility had the greater effect on early interactions, whilst set compatibility was more salient later on. A series of further studies is then reported, the aim of which was to investigate whether these effects were general across types and levels of interface, and other levels of task. Results mainly, but not entirely, indicated that they were.

Data from a more "ecologically valid" usability evaluation was re-analysed, to investigate whether the effects of consistency are important outside of artificial and tightly controlled experiments. Apparently they are — almost half of the difficulties encountered during users' early interactions with a commercially available word processor could be attributed to either set or rule incompatibilities.

Finally, a conclusions chapter outlines the implications of both the usability framework, and the distinction between set and rule compatibility. Possible directions for future research are suggested.

TABLE OF CONTENTS

PART 2: CONSISTENCY

LIST OF FIGURES

LIST OF TABLES

Chapter 7:Applying the theory to menu commands.

Chapter 8: Testing the guessability predictions.

Chapter 11: Investigating the effects of consistency at a higher level of an interface, with higher level tasks.

Chapter 12: Does consistency matter in "real" interfaces?

Chapter 13: Conclusions.

No tables

Acknowledgements

This thesis reports research conducted from 1990 to 1993, whilst I was employed as a Research Assistant at the University of Glasgow on SERC / DTI grant GR/F/39171/IED4/1/1109 — Measurement of User Interface Performance.

Thanks are due to my supervisor Steve Draper for comments on my work, for ideas raised during discussions, and for reading successive drafts of my thesis. I am also indebted to my colleagues in the Department of Psychology for advice and assistance given, in particular Eddie Edgerton and Judy Ramsey.

Much of the literature reviewed in this thesis was the work of Stephen Payne (University of Wales), and Thomas Green (MRC Applied Psychology Unit, Cambridge). On occasions when I have rung to request information regarding publications, or to ask for comments on my ideas, I have found them to be both courteous and helpful. Similarly, Andrew Howes (MRC Applied Psychology Unit, Cambridge), has also taken the time to comment on my work.

Coming, as I do, from an engineering background I initially had little idea about how to conduct behavioural research, until I completed an MSc in Ergonomics at the University of Birmingham. In particular I am indebted to Graham Johnson (then at Philips, Eindhoven, Holland, with whom I did my main research project), who introduced me to how empirical studies should be designed, run, and reported.

Finally, thanks to Joy Aked for her support and tolerance.

Chapter 1

INTRODUCTION

The topic of human-machine interaction (HMI) is increasingly coming to the attention of academia, industry, and the population in general. This is reflected in the growing numbers of academic journals, books, conferences, magazine articles and even television programs dedicated to HMI issues. This heightened awareness is not surprising — most of the Western world probably come across "high-tech" information technology based products everyday, either at work or in the home. These range from video recorders and microwave ovens, to sophisticated military equipment and the control banks in nuclear power stations.

The study of HMI encompasses a range of issues, from the social changes brought about by automation, to the effects of working with visual display units. The work reported in this thesis, however, is concerned with usability — how easily and effectively a particular user can perform at the human-machine interface — and with consistency. Loosely, consistency concerns whether or not similar types of task can be performed in similar ways at an interface, and this thesis is concerned with the effects of this on usability; their existence, nature, size, and practical importance. An empirical approach was taken, with an experimental methodology being employed throughout, to test the conceptual and theoretical ideas presented. The work reported is intended as a contribution to the HMI literature. Contributing to other related disciplines is not an explicit aim, although the methods used were originally developed in psychology.

This thesis offers two conceptual developments: distinguishing five components of usability associated with different parts of a user's learning curve, and distinguishing two types of consistency. The central portion of the thesis establishes the importance of the latter by demonstrating, experimentally, their independent effects on the former. The effects are large, but the experimental conditions are artificial. However, the last study reported offers evidence that the same consistency effects can also account for a substantial proportion of users' problems with a popular commercial product.

Firstly, a multi-component framework of usability is outlined: guessability, learnability, experienced user performance (EUP), system potential, and re-

usability. These five components embody an intuitively appealing distinction, describing how users' performance on a task might change with experience. Definitions of each of the components are offered, and their relationship to different sections of a notional learning curve illustrated.

An experimental methodology for measuring usability, and an analysis method for quantifying each component are then proposed. These involve monitoring performance on tasks over a series of trials, rather than on the one-off basis common to many traditional studies. Indeed, the validity of such studies is questioned on the grounds that it may not be clear as to which aspect of usability is being measured. Non-experimental evaluation techniques are reviewed, with suggestions as to how they could be used to measure the various usability components. A laboratory based experimental evaluation of a video cassette recorder is then reported. This illustrates how the proposed methodology and evaluation can be applied, and how the multi-component framework proved useful in classifying design problems with respect to their effects on different aspects of usability.

The next section of the thesis is concerned with consistency. A review of previous work shows how understanding of this concept has developed with successive treatments. This includes the work of Payne and Green, leading to the development of their widely used Task Action Grammar. Here, the concept is taken a stage further, with two separate types of consistency — set compatibility and rule compatibility — being identified and defined. Both components are concerned with matches (or mismatches) between aspects of the user's and the designer's mental contents — where the designer's mental contents are embodied in the interface. Set compatibility relates to whether or not a user and designer have the same ideas about which tasks can be thought of as being, in some way, similar. Rule compatibility, meanwhile, is concerned with whether a user's a priori expectations about the specific action-rules required for performing a task match those of the designer.

The ideas about usability and consistency are then brought together. Predictions are made about the effect of each type of consistency on the various components of usability. These assert that set and rule compatibility will each have different effects — rule compatibility being important during earlier interactions (guessability, learnability), with set compatibility being the more salient factor later on. It is predicted that both types of consistency will affect re-usability.

Assumptions underlying these predictions are then discussed. The approach to testing these predictions was empirical observation, rather than by looking at their fit or conflict with any particular theories of, say, learning or information processing. A series of five tightly controlled laboratory studies is reported. Three of these employ the methodology and analysis method outlined in the first section, whilst the other two investigate two separate types of guessability, and hence do not require performance profiles for individual tasks.

The studies looked at the effects of consistency in the contexts of different types of interface (menu based, iconic, and command driven). The effect of inconsistencies at different interface levels was also investigated. In the first four studies consistency issues were connected with the names (or representations) of commands, whilst in the fifth they were concerned with the medium through which users interacted (menus or keypad). Overall, results showed strong support for the predictions, generalised across different interface types and levels. A final study is then reported addressing the question of whether or not consistency matters in "real" interfaces — a pertinent question as the studies reported here, and those reported from the literature, were all based on users performing somewhat artificial tasks on specially created experimental interfaces. Set and rule incompatibilities in a commercially available word processing package were identified, and data from a more ecologically valid experimental study re-analysed to see if they had a major effect. Results suggested that almost half of the problems found could be attributed to inconsistencies, and thus that the issue might be of considerable importance in practice when designing for usability.

Bringing the ideas about usability and consistency together gave insights into both issues which might not otherwise have been possible. By applying the multi-component usability framework, it was possible to gain an understanding of the different effects of the two types of consistency, and to test the assumptions underlying the predictions made about their effects. Suppose that a one-point evaluation technique had been used. Depending on the level of experience of the experimental subjects, results would probably have suggested that only one of these types of consistency mattered. Even if effects were found for both, this would still not give insights into how the influence of each was experience dependent. Similarly, the different effects of set and rule compatibility on each usability component illustrated how these components could be independent — requiring separate consideration in evaluations.

PART 1: USABILITY

Chapter 2

A MULTI-COMPONENT FRAMEWORK FOR USABILITY

An important goal of HCI evaluation is to establish a framework for taking measurements which will effectively characterise the usability of an interface. This would aid comparisons of an interface against other interfaces or "benchmarks", and could support design improvements. The work of contemporary standards committees, such as the International Standards Organisation (ISO) (Brooke, Beven, Brigham, Harker, and Youmans 1990), and a number of earlier papers (e.g. Eason 1984, Shackel 1986) are steps towards this. Informally, usability refers to the costs to users of carrying out useful tasks with an interface (the benefits are sometimes referred to as the "utility").

The ISO (Brooke et al 1990) define the usability of an interface as "...the effectiveness, efficiency, and satisfaction with which specified users can achieve specified goals in a particular environment". This definition recognises that an interface's usability may vary between tasks (achieving a goal requires the completion of tasks in this context) and users. However, what is not explicit in the definition is that usability can vary within tasks and users. For example, the efficiency with which a task can be completed may vary according to the conditions under which it must be performed — what is easy under laboratory conditions may be more difficult in a "real life" setting. The user's frame of mind or degree of alertness may also affect performance. Perhaps someone who is tired will find a task more difficult then one who feels wide awake. These are factors which may vary from day to day, or hour to hour, and can be difficult to monitor. However, there is another important factor which varies over time within users which could be more easily monitored — experience. In this chapter a framework is presented which is designed to account for changes in an interface's usability with user experience.

In an earlier paper (Jordan, Draper, MacFarlane, and McNulty 1991) a three component usability model was presented describing how users' performance might be expected to improve with experience, before levelling out at an asymptotic level. In this chapter the three components are explained in greater detail, and two further components discussed. One is concerned with how easy it

is to come back to a task after a long gap between repetitions, and the other to relate measures of actual performance to a maximum performance achievable by an ideal user who has learned optimal methods. This extended five component usability framework can describe the phenomenon of "shells of competency" (Norman, Draper, and Bannon 1986), accounting for how users' performance can change over long time scales — the "asymptote" shifting up or down, for example, if new techniques are discovered, or a period of disuse leads to forgetting.

The five components are outlined in detail in the next section. Although derived from a very general concept of a learning curve, they are oriented towards empirical measurement and are not concerned with any particular psychological theory of learning. This emphasis on practical measurement is taken further in the next chapter, where methodologies for measuring the components of usability are discussed.

2.1 THE FIVE COMPONENT FRAMEWORK

The five components of the usability framework are described below. For each component the basic concept is introduced, examples are given of where it may be of particular importance, and a formalised definition, based on the ISO definition of usability is given. Whilst it is accepted that for any given interface the components may not always be independent, they are at least conceptually distinct.

2.1.1 Guessability

This is a measure of the cost involved in using an interface to perform a new task for the first time — the lower the cost the higher the guessability, where cost may be measured in terms of, say, time, errors, or effort. Guessability is likely to be important with interfaces that have a high proportion of one-off users, for example, fire extinguishers, door handles on public buildings, or public information systems. Also, a lack of guessability could put users off an interface which in the long run might be comparatively easy to use. For example, a customer may choose to buy the stereo which was easiest to use at the first attempt in a shop, which may not necessarily be easier for the experienced user.

Guessability is of less importance in situations where procedures are initially demonstrated to the user, or for interfaces which will only be used by experts after long training. This might include, say, military equipment, and aircraft controls. However, note that emergency diagnosis and recovery procedures should still be guessable for pilots — even though they may be experienced generally, there may still be rare tasks which they have never seen before. Another important case is whether a user with wide experience of a particular interface can find a new function when needed. Menus, for example, are one way of addressing this type of guessability.

> Guessability: The effectiveness, efficiency and satisfaction with which specified users can complete specified tasks with a particular interface for the first time.

2.1.2 Learnability

This refers to the cost to the user in reaching some competent level of performance on a task, but excluding the special difficulties of completing the task for the first time. The issue of making judgements about what can be described as "competent" in a particular context is discussed in the next chapter. A highly learnable interface would be one where a task was instantly memorable once the method had been shown to the user. Conversely, interfaces which cause "interference" with user expectations are likely to be un-learnable. For example, an interface which relied on perverse command names might score a high penalty here. Interfaces which rely on recognition might be expected to be more learnable than those which rely on recall. The user of a recognition based interface (for example a graphical, or menu driven interface) can operate effectively without having to retain detailed information about the interface (Mayes, Draper, McGregor, and Oatley 1988).

Learnability may be particularly important where a user is to be self-taught with an interface, or where training time is short. For example, temporary secretaries may be introduced to new word processing packages on a fairly regular basis. Clearly, significant amounts of working time will be wasted if they do not reach a reasonable standard of performance fairly quickly. Learnability will be of less importance where training time and resources are more plentiful; again a pilot learning to fly an aircraft would be an example of this.

The term "learnability" has already appeared widely in the usability literature. However, its meaning has not always been clearly expressed, and different practitioners appear to mean different things by it. For example, Payne and Green (e.g. 1986) use the term with reference to completing new tasks for the first time (here referred to as guessability), whilst others (e.g. Shackel 1986 and 1991) use the term more loosely, to refer to performance by any user who could not be deemed experienced. Indeed, the idea that there is a distinction between usability for an experienced user, and usability for one with less experience is not uncommon. However, the distinction is usually couched in rather vague terms, with little attention to the guessability / learnability distinction or the criteria by which a user is judged to be experienced or expert. For example, many studies in the literature, described as studying new users, either discard the time to complete tasks for the first time, or give subjects special help at this stage, thus omitting guessability. (Perhaps this is because this appears to have been standard practice in many areas of learning psychology).

> Learnability: The effectiveness, efficiency and satisfaction with which specified users can achieve a competent level of performance on specified tasks with an interface, having already completed those tasks once previously.

2.1.3 Experienced user performance (EUP)

This refers to the relatively unchanging performance level of experienced users. Intuitively, it may appear that practiced users' performance will level off to an asymptotic level, although for behaviours following the power law of practice (Card, Moran and Newell 1983) this is not an exact definition. However, although experienced performance may vary over long timescales, it may be expected to remain comparatively steady when compared to performance during the learnability phase. This component is often what is meant by practitioners when they use the term "usability" (e.g. Bennett 1984, Eason 1984, Shackel 1986).

EUP will be comparatively important in situations where constraints on training time and resources are few, but where it is important that the experienced user makes few errors. Again, flying an aircraft would come into this category, as would driving a car, or perhaps using a specialist software package, for example for computer aided design (CAD).

EUP: The effectiveness, efficiency and satisfaction with which specified experienced users can achieve specified tasks with a particular interface.

2.1.4 System potential

This represents an idealised maximum user performance, and, as such, is an upper bound on EUP. Unlike the other components, which represent properties of the interaction between the user and the interface, system potential is a property of the interface alone, and does not vary either over time or between users. In the context of a computer based interface, system potential may be represented, say, by the number of keystrokes or mouse movements required to complete a particular task.

Clearly there will be instances where there is a gap between system potential and EUP, perhaps due to a user learning a non-optimal method of task performance, or simply not having the ability or opportunity to push task performance to its limit. For example, many mouse and menu interfaces have alternative accelerator key options for performing many tasks, however, it is often the case that experienced users will not bother to learn these, but stick with non-optimal method of command selection from menus. An example of where a user is usually prevented from exploiting full system potential is with automobiles. Although market research suggests that a major selling point for vehicles is their performance, in terms of, for example, top speed, the roads would be extremely hazardous if drivers were to exploit this on a regular basis.

Over longer time scales EUP may move nearer to system potential, perhaps creeping up due to a gradual improvement in some ability secondary to the main task. For example, a user might be able to enter commands more quickly due to gradually improving typing speed. However, sometimes step jumps in performance might occur, perhaps due to the user discovering some new command or technique. Norman, Draper and Bannon (1986) refer to this phenomenon as "Shells of competency". Returning to the menus vs. accelerators example from the previous paragraph, it might be expected that a user who started using accelerators rather than the menus would experience a sudden performance improvement. How close users' performance gets to system potential is likely to be dependent upon the complexity of the optimal method, and their motivation to

learn it — some users may be content to remain at what they regard as a reasonable level of performance.

System potential will be important in cases where it is the limiting factor on EUP. For example, with a command driven interface the time taken to type long command strings may be irritating to a user, even if they no longer have any problems remembering the contents of the strings. It will be of less importance when EUP is limited by other factors. For example, the reason it may take several hours for a student to word process an essay is likely to be because of limits on thinking and typing speed, rather than anything to do with the time that the package takes to respond to keystrokes.

> System potential: The optimum level of effectiveness, efficiency and satisfaction with which it would be possible to complete specified tasks with an interface.

2.1.5 Re-usability

This component, similar to what Shackel (1986) refers to as re-learnability, concerns the possible fall-off in user performance after a long gap, perhaps due to forgetting how the task was done, or, say, forgetting where to find a particular menu stored command. Re-usability is possibly concerned with more complex issues than the other components. Firstly, the idea of a "long gap" is somewhat vague. An implicit assumption associated with learnability and EUP is that users are repeating a task on a fairly frequent and regular basis. The "gap" associated with re-usability can perhaps most usefully be thought of as a length of time significantly greater than the usual gap between task repetitions.

Clearly, re-usability may be dependent on the length of this gap, although there may be some tasks with which even a short time away brings the user back to the start of the learning curve, whilst with others users may remain fairly near EUP even after a long gap. For example, it is possible that many students would now have no idea about how to perform a long division, even if they were proficient at this when in school. Conversely, it is widely held that one who has learned to ride a bicycle never loses this ability. Presumably re-usability will follow a curve, but, there is no guarantee that its shape will be similar to the original learning curve.

For example, a task may initially prove very difficult to come back to, however after the user's memory has been jogged, performance may rapidly return to EUP.

Re-usability is likely to be important in situations where users may only use an interface in intermittent "bursts". For example, a researcher may use a statistics package fairly regularly for a week or so after completing a study, but may then have a gap of a couple of months before using it again after a further study has been completed.

> Re-usability: The effectiveness, efficiency, and satisfaction with which specified users can achieve specified tasks with a particular interface after a comparatively long time away from these tasks.

Each of the components of usability is loosely associated with a different section of a notional learning curve. An example of such a curve is illustrated in figure 2.1.

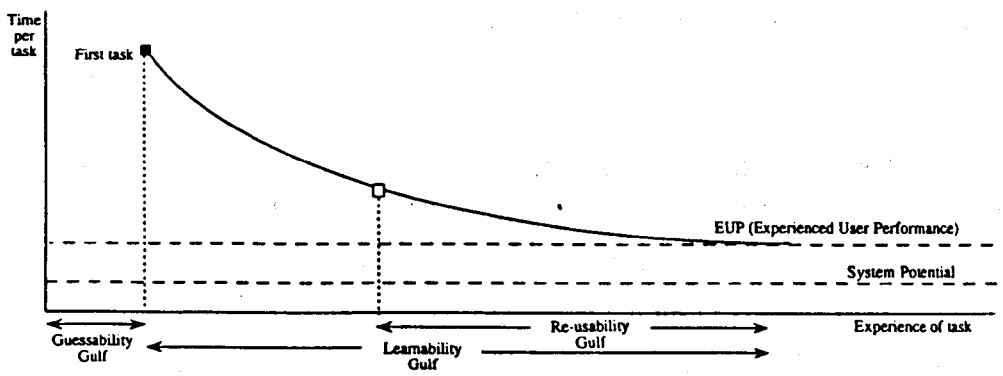


Figure 2.1. Idealised learning curve illustrating the five components of usability.

2.1.6 Implications for usability evaluation, and design

Perhaps the most obvious corollary of the five component framework, is that evaluations should be designed with clear ideas of the component of usability which is of interest. For example, if interested in guessability subjects should not be given the opportunity to practice the experimental tasks beforehand, whereas if EUP were of interest subjects should be well practised in the experimental tasks. This may cast doubt on some of the "traditional" approaches to usability evaluation, where claims about usability are often made on the basis of one-off

measures of performance taken after apparently arbitrary practice opportunities. If, for example, users were given, say, a twenty minute practice session, during which they were free to explore an interface, including, possibly, the experimental tasks, it might be difficult to know what their level of experience on a particular task was when the experimental session commenced. Without knowing this it would not be clear which component of usability was being measured. This practice is commonly reported in the consistency literature (discussed further in chapter 6).

Other types of study have had more closely controlled practice opportunities, but then still made claims for usability as a whole on the basis of one-point measures. An example, from this author's own work, was an investigation of a remote control for operating an in-car stereo (Jordan and Johnson 1991). Subjects had not used the remote prior to any experimental measures being taken. These measures, then, would be representative of guessability. However, because no distinctions were made between different usability components, performance was simply taken as representing the device's "usability", without qualification for the effects of experience. Ideas about methodologies for usability evaluation are discussed in the next chapter.

From the designer's point of view the framework should aid clarity in defining usability goals. Indeed slogans such as "design for usability" have little practical meaning unless the question of who an interface is to be usable for is addressed, and an important part of this is the expected level of user experience. For example, if designing a piece of specialist software, to be used by highly trained users on a regular basis, it may not be worth the effort and expense of designing a highly guessable, learnable, or re-usable interface provided that high EUP is supported. In practical terms this might mean, say, designing a command key based interface, rather than one with menus. Conversely, an interface to, say, a walk-up-and-use health information system should support high guessability, but need not support high learnability or EUP.

Of the many properties of an interface that influence usability, including, for example, feedback, layout, visibility, and consistency (Jordan 1992), some may have greater effects on a particular component of usability than others. Say, for example, that in an imaginary interface a high degree of visibility would support high guessability, and that an uncluttered layout was essential for high EUP. The designer may be faced with the predicament of either displaying lots of command

names, menu headings, icons etc. on the screen to aid the first time user, or having them hidden at another "level" of the interface to avoid clutter. Even if the designer cannot cater for all potential levels of experience, at least the framework creates a frame of reference for any trade-offs which might have to be made.

2.1.7 Classifying users

A fairly common theme amongst many HCI practitioners, is to make a distinction between "expert" and "novice" users (e.g. Fisher 1991, Streitz, Spijkers, and Van Duran 1987). The multi-component usability framework appears to support the possibility of classifying users with respect to experience. When first reporting the idea of the guessability / learnability / EUP distinction (Jordan, Draper, MacFarlane and McNulty 1991), a three way user distinction was proposed:

> Naive users: Users who have not yet completed a particular task with an interface.
>
> Novice users: Users who have completed a particular task with an interface one or more times, but not enough times to reach a steady "asymptotic" level of performance.
>
> Experienced users: Users who have completed a particular task with an interface enough times to reach a steady "asymptotic" level of performance.

When considering system potential and re-usability, additional user classifications could be:

> Expert users: Users performing at the limit of a system's potential on a particular task. (This would be a sub-category of experienced users).
>
> Returning users: Users returning to a task with an interface after a long period away.

Note that these user classifications, like the usability components, are all defined in relation to a particular task. When studying users operating with the UNIX system Draper (1985) noted that individual users would often obtain "expert"

status on limited numbers of tasks, whilst possibly being "naive" or "novice" in relation to others. Thus trying to give general labels to a user may prove inappropriate and unhelpful. What is more important is to have an idea of users' working patterns. For example, if the users in Draper's study rarely used commands other than those with which they had obtained expertise, then system potential, as the limiting factor on EUP might be the only component of great importance. Conversely, if they were occasionally likely to use other commands on a one off basis guessability would also be important.

2.2 PROSPECTS FOR GENERALISATION

A possible criticism of usability definitions, including the framework outlined here, is their narrowness — the reference to "... specified users performing specified tasks...". So, how optimistic is it reasonable to be about the possibility of making generalised statements about the usability of any given interface? The answer is likely to be dependent both upon an interface's user population, and the range of tasks which users are apt to perform with it.

2.2.1 Inter-user variability

Baxter and Oatley (1991) identified three types of user experience which may affect performance when using an interface:

> Task domain experience: Gained from experience of performing a task, irrespective of the interface with which it is performed. So, for example, experience of writing the word "computer" will be helpful whether the word is being written by hand, typed, or word processed.
>
> Operating environment experience: This experience can be gained through familiarity with a particular "type" of interface. For example, the Apple Macintosh has conventions that are used in many different types of application program. Commands for saving work and printing can usually be found on a menu headed "FILE", irrespective of whether the application is, say, a word processor, or a statistics package.

Application program experience: Gained from experience of using a particular application program, perhaps in a different operating environment. For example, a particular graphics package may employ similar icons regardless of the operating environment in which it is run.

Differences between users with respect to each of these experience types can cause marked differences in performance. For example, in a comparison of two different Macintosh based spreadsheet packages, Baxter and Oatley (1991) found that the level of task domain experience which first time users brought with them far outweighed any differences between the packages as a factor affecting performance. Thus, in this case, the usability of the machine was very much user dependent. Presumably the other types of experience could have equally dramatic effects on the usability of interfaces for some tasks.

However there will be cases where the user population is fairly homogeneous as regards the three experience types. For example, few of the potential users of a fire extinguisher are likely to have operating system or application program experience (in this case the two are essentially the same as the fire extinguisher has only one function), although some might have task domain experience (this means that they may have put out a fire, for example a bonfire, an open hearth fire, or a chip pan fire). Thus, whilst users know that to complete their task they must direct the contents of the extinguisher onto the flames, they may not know how to release the contents from the extinguisher. Therefore, there appears no obvious reason to expect a great deal of inter-user variability. Even in an apparently clear cut situation such as this however, there could be "hidden" factors which have a big effect. For example, if a user had some knowledge of valve release systems gained in another context this might be helpful.

2.2.2 Inter-task variability

The inter-task variability of an interface's usability is likely to be dependent on the range of task types which it supports. A public information system, for example, may support only one type of task — accessing information. Further, the method used for accessing each piece of information may be fairly similar. It might be expected, then, that the usability of an interface to such a machine might be fairly uniform across tasks.

With more complex systems the range of task types will be wider and thus inter-task variability will probably be greater. For example, a statistics package which is easy to use for performing calculations, may not necessarily be usable, say, for creating graphs to represent the data. Probably the majority of modern application programs do tend to support a wide variety of tasks. Thus, a degree of caution shall be necessary before trying to make generalisations across an interface.

2.2.3 Experience on a task vs. experience on an interface

Another issue is whether experience can be thought of in terms of experience with an interface as a whole, or whether it is only useful to consider it in task specific contexts. If the former were true, then it might be possible to obtain performance profiles of, say, total number of interactions with the interface versus time to complete each interaction, and then derive measures for each component of usability for the interface as a whole.

The feasibility of such an approach — treating knowledge of an interface as a whole as a single variable — is likely to be dependent on the prospects for transfer of knowledge from one task to another. If the interface were designed in such a way that performance on any particular task could be influenced, as much by experience of other tasks, as by previous experience on that task itself, then this approach may seem reasonable. However, if this is not the case, it seems doubtful that such an approach would give meaningful results.

Overall then, it is probably not sensible to attempt to quantify the various components of usability for an interface as a whole. Rather it may be more constructive to think in terms of how experience of one part of an interface may affect the usability of another part. Consider, for example, a word processor employing a mixture of menus and command keys. It might be expected that experience of activating menu based commands might help the user perform a new task which also relied on a menu command — thus making this task more guessable. In this situation, the user might be able to transfer knowledge about, say, how to open a menu and select a command. It is also possible that they might have seen the command required for their new task in the course of previous interactions. However, this previous experience may not enhance the task's guessability if the new command required had to be activated via the command

key. Indeed menu experience could diminish guessability in this case as the user might decide to try some menu commands which, here, would be inappropriate. Studies addressing how experience of some tasks may affect the guessability of others are reported in the next part of this thesis.

2.3 ARE THE COMPONENTS OF USABILITY INDEPENDENT?

Whilst the components of usability outlined in this chapter might be conceptually separate, they may not always be independent. It may be, for example, that a task which has a high associated re-usability might also have high learnability. Indeed, it might be expected that if a designer has kept to a set of sound usability guidelines or design techniques (e.g. Thimbleby 1991, Booth 1989) that every aspect of usability would benefit. However, this need not always be the case. It may be that some aspects of design will affect the components in different ways — the second part of this thesis reports on how consistency can affect the different components differentially.

Similarly, differing user strategies may also affect whether the components are independent. Users who take a trial and error approach to a task may, for example, find a system fairly guessable on the basis that they will eventually "hit the right button". However, they may not have improved much by the time they reach EUP. Conversely, users who try to work from some sort of system model might have difficulty early on, if their models were incorrect to start with, but may perform well at EUP if they have, by then, adopted the correct model.

Whilst comments in this section of the chapter may indicate caution about the prospects for generalisation across tasks and users, it is perhaps unnecessary to be wholly pessimistic about the prospects for making meaningful statements about an interface's usability. Good evaluations will employ reasonably representative user populations, and a range of tasks representative of the types of interaction techniques applicable to an interface. It should be possible, then, to gather data from which meaningful comparisons between interfaces can be made, or by which interfaces could be compared to some sort of benchmark standard. Methodologies for evaluating interfaces with respect to the various components of usability are discussed in the next chapter.

2.4 SUMMARY

A five component model of usability has been outlined, accounting both for changes in a user's performance with experience, and the limitation on performance imposed by the interface. Building on the ISO definition of usability, these components refer to usability in the context of a particular user performing a particular task, although, with good evaluation practice, it may, hopefully, be possible to generalise across tasks and user groups.

The components are conceptually separate and potentially independent. This throws some doubt on the validity of one-point usability evaluations.

Chapter 3

HOW THE COMPONENTS OF USABILITY CAN BE MEASURED

Having outlined a multi-component framework for usability, and questioned the validity of some one-point usability evaluations, it would be constructive to put forward ideas for more comprehensive usability measurement. This chapter discusses possible methodologies and analyses for quantifying the five components of usability. There are two main sections to the chapter. In the first, possible experimental methodologies and analyses are considered, and a criteria based approach outlined which formed the basis for analysis in the studies subsequently reported in this thesis. In the second, some non-experimental methods are introduced. This is to demonstrate that the multi-component framework is not only useful in the context of experimental evaluation, but rather that alternative, perhaps less costly, methods can also be used to gain information about the various components of usability.

3.1 EXPERIMENTAL TECHNIQUES

This section starts with a review of a laboratory based technique and "semi-formal" analysis method for quantifying guessability, learnability, and EUP which was originally presented to the Ergonomics Society conference (Jordan and O'Donnell 1992). Other possible experiment based analyses are also discussed, and a practical, less complex, analysis method is proposed.

3.1.1 Jordan and O'Donnell's "semi-formal" analysis

This was an experimental approach, based on the idea that in order to quantify guessability, learnability, and EUP, it was necessary to obtain a series of points to which it was possible to fit a curve with a well defined starting point, and some sort of asymptote (or at least a point where the slope starts to level off). This means implicitly that it will be necessary for experimental subjects to repeat trials (comprising a task or series of tasks) several times, so that their change in performance with practice can be observed. Performance, for example, in terms of

time on task, or errors, can then be plotted against trial number. Having obtained such a plot, the requirements of the analysis are to define a starting point, from which to derive guessability, an asymptote, to represent EUP, and a point at which performance could be said to be reasonably close to this asymptote, from which to derive learnability.

It was decided that the start point of the curve should be taken as the point at which the first trial was completed. This point was chosen as it was directly measurable, and did not rely on any assumptions about what the theoretical shape of the curve may have been prior to completion of the first trial (if indeed this is a meaningful notion).

For their analysis Jordan and O'Donnell decided to represent performance in terms of cumulative trials against cumulative time on task. This gave plots of a similar shape to that illustrated in figure 3.1. The slope of such a curve increases with improvements in performance. When steady performance is reached the slope becomes linear. In order to gain values for EUP and learnability, it was first necessary to choose a criterion for deciding whether or not a subject's performance had reached a steady level during the experimental session. In terms of this type of performance profile, this would be represented by a linear section of a curve.

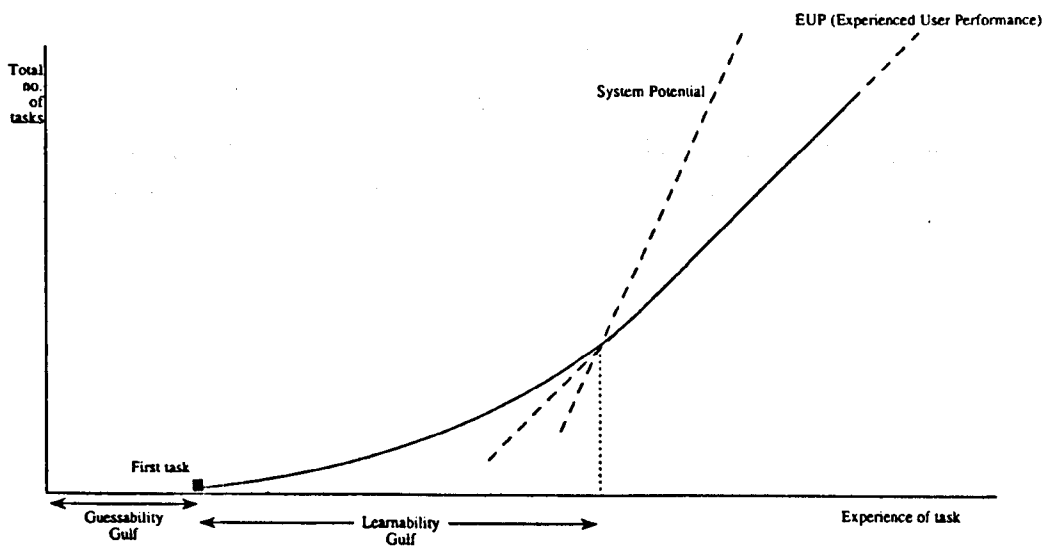


Figure 3.1. Idealised performance profile of the form used by Jordan and O'Donnell (1992) in their quantification of guessability, learnability and EUP.

By fitting a linear regression equation to the last few points collected from each subject's data, Jordan and O'Donnell checked whether, by the end of the session, the relationship between points on a performance curve could be adequately described by a straight line. The minimum number of points to which it is useful to fit a straight line is, in this context, three (any two points, considered in isolation from the rest of the curve, can be said to have a perfect linear relationship). However, it was decided that four points should be used for the analysis, to reduce the risk of any freak data points "throwing" the results. The adequacy of a straight line fit to a set of data points is described by the co-efficient of determination (R-squared), associated with the linear regression equation (Daniel 1977). Low values indicate that the closest linear fit is not really an adequate description of the relationship between the points. However, a high R-squared value indicates that a linear descriptor is adequate, and that the four points can be taken as lying on a line sufficiently straight to represent an area of steady performance. (Jordan and O'Donnell took an R-squared value of 0.995 to be "high"). EUP, then could be taken as the mean level of performance over the trials represented by these points. Note that analysing the last four points for linearity means that it is the equality of performance on the last three trials which is being investigated — four points gives three line segments each representing one trial.

The particular form of curve used here (cumulative trials vs. cumulative time on task) was chosen as steady performance would be represented by a linear segment of any slope to which it would be possible to fit a linear regression equation. With other forms of curve (for example, trial number vs. time on task), steady performance would be represented by a flat portion (see figure 3.2). Unfortunately, linear regression tests cannot be used to test for flatness, hence other forms of plot would not be useful with this type of analysis.

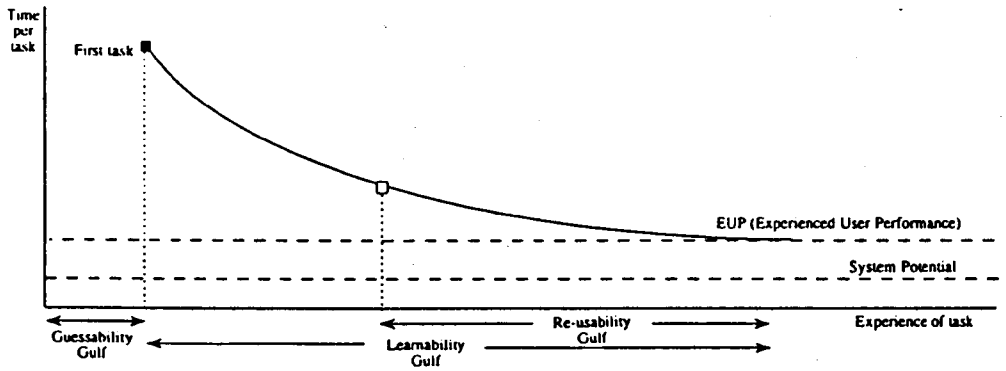


Figure 3.2. Idealised performance curve illustrating time on task vs. trial number.

In order to obtain a value for learnability, a criterion was decided upon for determining a point at which each user's performance curve could reasonably be said to have started to level off. It had previously been suggested (Jordan et al. 1991) that being within 95% of EUP was a reasonable criterion, and it was this which was adopted for the analysis. Hence, if the linear section of the curve suggested that a user was completing, say, 100 trials per hour at EUP, then the first trial at which the rate of task completion was greater than or equal to 95 trials per hour, could be taken as a reasonable estimate of when a steady level of performance was reached. A value for learnability, then, could be taken from the appropriate trial number, or the cumulative time taken to reach this trial number minus the time to complete the first trial (this would be covered by the guessability measure).

This analysis method was applied in the context of the evaluation of a video cassette recorder (VCR). The study is reported in Jordan and O'Donnell (1992), where it is concluded that the analysis is "...simple enough to be practical, whilst affording a degree of formality in the assessment of guessability, learnability, and experienced user performance."

Whilst developing the analysis, Jordan and O'Donnell initially considered three further options: fitting an equation, judging the data by eye, and testing for significant differences between performance from one trial to the next. These rejected options are discussed below.

3.1.2 Fitting an equation

An initial option was to try to fit a mathematically defined curve to some measured points, in the hope that the level and start point of an asymptote could then be calculated — however, two major problems arose. Firstly, there is no one generally accepted model covering all performance curves (Mazur and Hastie 1978), yet trying to fit a curve to data points without a model is (in the words of Newell and Rosenbloom 1981) "... a notoriously black art." Secondly, the models most frequently suggested in the literature, for example exponential, hyperbolic, or the power law of practice (Card, Moran, and Newell 1983), tend to be fitted to far greater numbers of points (often tens (e.g. Card, English and Burr 1978), hundreds (e.g. Kolers 1975), or thousands (e.g. Crossman 1959)) than can conveniently be obtained from a usability study, where experimental tasks tend to be comparatively complex, and take many seconds each. Having only a few points may cause problems in this context, as usually it is the earliest of the empirically measured points that are most likely to deviate from any fitted equation (Newell and Rosenbloom 1981).

3.1.3 Judging the data by eye

The major problem with judging for an asymptote and its start point by eye, is that judgements made are likely to be too subjective, and unreliable. The semi-formal analysis can provide a set of constant criteria for deciding whether or not a curve has become linear, and the point at which linearity has been reached. Without such criteria there are likely to be inconsistencies between how two separate analysts judge the data, and even between the way a single analyst views data on separate occasions. This introduces a potentially large source of error to any values assigned to usability components.

3.1.4 Testing for significant differences between trials

This approach was used by Mohageg (1991) when comparing the usability of two different graphical interfaces. The idea is to test whether or not the performance of a group of subjects has improved to a statistically significant degree from one trial to the next. If there is a statistically significant difference, then performance is

regarded as still improving, however, if there isn't then it is regarded as having levelled off.

The problem with this method is that the likelihood of a difference being statistically significant is liable to be dependent on the subject sample size. Consider, for example, a mean difference in time on task between two consecutive trials of 20 seconds. If the sample size were, say, sixteen then this difference might prove statistically significant, whereas with a sample of, say, eight, it may not. Again, then, a potential source source of inconsistency would have been introduced into the analysis.

3.1.5 Quantifying system potential and re-usability

The analysis techniques discussed so far make no provision for quantifying system potential or re-usability. By definition system potential means error free performance, but how can it be quantified in terms of, say, time on task? An experimental measure might be gained from measuring the performance of an over-practiced expert, perhaps the experimenter, who may know an interface "inside-out" by the time he or she has monitored a group of subjects performing the experimental tasks. This approach was taken in the study reported in the following chapter.

In order to measure re-usability experimentally, it will be necessary to ask subjects to return to the task after a period away. For example, having completed a series of trials during a first experimental session, subjects might return a day later to attempt another trial. Re-usability values could then be derived from performance on this trial.

3.2 CRITERIA BASED ANALYSIS

Whilst Jordan and O'Donnell's (1992) analysis method introduces a degree of formality, it is costly to apply in terms of experimenter time and effort. It would be preferable to have an analysis which, whilst offering a degree of formality, could be applied more quickly and efficiently. Alternative analyses have been discussed, but each found to be potentially flawed. In this section, the possibility of using criteria based analyses is discussed.

The idea is based on the approach of Gilb (1981), which advocates setting usability specifications to guide the design and evaluation of an interface. For example, a specification may state that for an interface to be acceptable, 90% of users should be able to complete task X within 30 seconds. In the context of measuring the components of usability, this could be adapted by specifying criteria which define, say, which users can be regarded as experienced, or what level of performance represents having learned a task. For example, it might be decided that if a user had completed a task, say, ten times, they could be classified as experienced on that task. Similarly, performing a task in less than, say, a minute could be taken to represent a competent level of performance, so that users could be said to have learned that task. If this approach were used, EUP might be taken as mean performance on subjects' tenth trial, and learnability defined by how many trials it took each subject to complete a task in less than a minute. This was the approach used throughout the studies reported in this thesis.

The way in which criteria are set may be dependent on the context of the study. Most of the studies reported in the next part of this thesis are experiments on specially created interfaces, for the purpose of investigating how different properties of a design affect usability. What was of interest, then, were comparisons between performance on different types of task. Here criteria were set post-hoc in order to facilitate meaningful comparisons. So, if by the end of an experimental session users tended to be completing tasks of all types in less than, say, five seconds, it would have been helpful to regard this as a competent level of performance. This meant that learnability could be quantified according to how long it took to reach this level on each type of task, and meaningful comparisons made. Similarly, if performance on all tasks appeared to have levelled off after, say, ten trials, performance from here on could be used taken as representative of EUP.

When considering the design of an interface, it may be more appropriate to set target criteria, a priori. For example, "this interface is learnable if 90% of users can complete tasks in less than one minute after five repetitions", or "the interface supports an acceptable level of EUP if 90% of users can complete tasks in less than 30 seconds by their tenth repetition". This, then, is in accordance with Gilb's (1981) ideas of setting usability specifications, and could support the approach of practitioners such as Whiteside, Bennett, and Holtzblatt (1988), who advocate that the use of specifications is central to design for usability. These specifications may

be set in accordance with the needs of, say, a job, or a training program. For example, a manufacturing company's profitability might depend on a computerised-numerically-controlled (CNC) milling machine operator being able to turn out, say, ten components per hour, or a company might be able to give its secretaries a couple of days to learn a new word processing package, during which time they might expect them to use some of the basic commands about ten times each.

Another potential problem with Jordan and O'Donnell's approach, is that their analysis may make values gained for learnability dependent on EUP. Consider two curves, illustrating performance on similar tasks, but with different types of interface, each curve being a plot of performance over twelve trials. Imagine that they had identical profiles up until, say, the sixth trial, where one started to become linear, the other not reaching linearity until, say, the eighth trial.

With Jordan and O'Donnell's analysis, the second of these would have a higher (faster, less costly) EUP than the first, which is sensible. However, because the second's profile takes longer to stabilise, the analysis would assign a worse learnability value to this interface. So a comparison between the two interfaces would suggest that, although the second had a higher EUP, the first was more learnable — implying some kind of trade off between the two. Of course, this isn't really the case, as learning proceeded at the same rate, but rather the second interface has been "penalised" in terms of learnability, simply through supporting a higher level of EUP. With a criterion based approach this anomaly would not occur, as both profiles would reach the criterion level together (provided this was lower than the first curve's asymptote), giving equal learnability ratings.

Because criteria based analyses require explicit statements to be made about the level of performance which is accepted as representing competence, learnability values are meaningful in their own right. This is not really the case in Jordan and O'Donnell's analysis where they must be interpreted in the light of the associated level of EUP.

Another limitation of Jordan and O'Donnell's analysis, is that it was specifically designed with time on task in mind as the measure of performance. It seems unlikely that such an approach would work with, say, errors as the measure of performance. If error rates were low, say moving towards zero at EUP, then a linear regression analysis would not work, as even a curve of the form chosen for

their analysis would become vertical, making linear regression analysis impossible. Criteria based analyses, however, can be used with any performance measure.

Criteria based analyses are also free of the problems inherent in the analysis methods rejected by Jordan and O'Donnell. Unlike equation fitting they do not rely on any assumptions about the shape of users' performance curves. Because fixed criteria are explicitly set, they provide data which is less subjective, and more reliable than judgements made by eye. Nor will values gained be dependent on the number of subjects, a risk of testing for significant differences between trials.

A possible criticism of criteria based analyses is that, unlike Jordan and O'Donnell's analysis, no mathematical check is carried out to see whether user's performance has become steady. Clearly, when setting criteria post-hoc, it may be worth looking at a few users' performance profiles, or a profile of mean performance, to check that performance seems fairly steady by the time the criterion number of trials is reached. Where criteria are being set as part of a design specification, the issue of steady performance may not be important. Consider, for example, that specifications state that an interface must support task completion in under, say, ten seconds, for a user who has performed this task ten times or more. If by the tenth trial performance has reached this level, it doesn't really matter if performance hasn't levelled off by then — the specifications have been satisfied. Indeed, any subsequent improvement in performance would be a bonus.

3.3 NON-EXPERIMENTAL TECHNIQUES

The analysis techniques and methodologies reviewed so far all rely on an experimental approach to evaluation. The experiment is a thorough technique which, if properly designed and controlled, can provide comparatively unambiguous data. This makes it particularly suited to studies, where, (as with those reported in the next part of the thesis) specific effects or manipulations are being investigated. However, there are situations where experiments may be less appropriate.

The main problem with experiments is the cost to the investigator of designing and running them. If the investigator has to be in attendance during each subject's experimental session, this can be very time consuming. This cost is likely to be accentuated where subjects are repeating a number of experimental trials, as would be the case when using the methodology advocated here. Design of an experiment may also require specialised knowledge which many investigators may not have — for example, how to balance across different conditions, how to control for experimental noise. Another potential problem, is that effects which appear important in tightly controlled laboratory based experiments may not always transfer to other more natural situations, perhaps becoming swamped by other factors. In this section some non-experimental evaluation techniques are reviewed. Firstly, other types of empirical technique are discussed, followed by an introduction to non-empirical techniques.

The studies reported in this thesis all had experimental methodologies. However, the aim of this section is to demonstrate that the usability framework employed is not only useful in the context of experimental evaluations, but that the components of usability could be investigated using a variety of techniques. Along with brief descriptions of each of the methods introduced, examples are given of how information about the various components of usability might be gleaned.

3.3.1 Empirical techniques

Jordan (1993) reviews seven empirical techniques for usability assessment, six of which are non-experimental. In this section brief descriptions of each of these six are given, with suggestions as to how each method might be used to capture information about the individual components of usability. Because these methods gather data via users, rather than by directly monitoring performance, they can give important insights into the effort and satisfaction components of usability. These could not be obtained purely from performance measures. So whilst, performance measures may rate two interfaces as being equally usable for a task, users may rate one of the two as being more difficult to use than the other. This may reflect a higher level of user effort being associated with one of the interfaces. Similarly, asking users, say, how much they enjoyed performing tasks may give data useful in gauging user satisfaction.

3.3.2 Focus groups

The focus group (O'Donnell, Scobie, and Baxter 1991) is a group of people brought together to discuss a particular issue. Discussion could centre on, say, users' experiences of using an interface, or ideas about the sorts of characteristics and functionality they would like a machine to have (requirements capture).

The leader of the group has an agenda of questions, which he or she can raise, for the participants to discuss. The aim is that participants should be allowed free discussion amongst themselves, but that this discussion should centre on the particular topics on the agenda. By listening to the discussion, and taking notes, an investigator can glean information about an interface.

To use a focus group for quantifying the components of usability it would be necessary to adapt the agenda appropriately. This might mean, say, including questions such as, "Are new tasks easy to complete at the first attempt?" (guessability), or, "If you come back to tasks after a long gap can you usually remember how they are done?" (re-usability). Data gained from a focus group will be mainly of a qualitative nature, although quantification of the type "X users agreed that this was a problem" is possible. Because questions can be fairly broad, rather than asking about specific tasks, the data may provide an overall picture of an interface with regard to each usability component.

3.3.3 Think aloud protocols

This method involves asking a user to give some form of commentary about what they are doing and thinking whilst using an interface, in response to prompting from an investigator. Meanwhile the investigator observes, listens, and takes notes.

The main advantage of the think aloud protocol is that it helps the investigator to identify not only what kinds of problems users are having, but also why they are having difficulties. During the think aloud session the users may be set specific tasks to perform (e.g. Jordan and Kerr 1993), or may simply be asked to explore the an interface freely. The former approach is useful for identifying specific bugs, whilst the latter may help to explain why users use some parts of an interface yet ignore others.

When investigating the components of usability it may be most appropriate to set the users specific tasks. By asking about previous experience with each task, it should be possible to get an idea about which component of usability is being measured. For example, if a subject were to say, "I have been doing that task virtually every day for the last month", then it might be reasonable, to expect that EUP were being measured, whereas had he or she said "This is my first attempt at a task", then it would be guessability. Because the subject will be talking during a think-aloud session, this may interfere with task performance, perhaps making this method unreliable as a source of performance data. However, because the content of subjects' verbalisations may be useful in explaining why any problems occur, data gained could prove useful in the formation of design solutions.

3.3.4 Incident diaries

These are mini-questionnaires which users are asked to keep with them whilst using an interface. The idea is that they make an entry whenever an "incident" of some kind occurs. This might mean, for example, when a user makes an error or has difficulties on a task, or when they are confused by a system action. Typically (e.g. Jordan 1992a) a diary might contain questions about what types of problem occurred, how troublesome these problems were to the user, and if and how they were resolved.

Incident diaries are useful when relatively infrequent problems occur, and the investigator cannot be there to observe them. The method is cheap in terms of investigator time and effort, as having decided on a set of questions, these can then be copied and sent to many users. However, it is not always clear whether respondents are describing problems accurately — indeed they might sometimes lack the technical vocabulary to do this. It is also important that the diaries are short and clear, so that they can be completed quickly and easily — otherwise users may not bother with them.

Reliably completed incident diaries can be a useful guide to an interface's usability. Any task mentioned at all by the user, might be regarded as being un-guessable. However, it is those which re-occur which may be poor as regards learnability, or (depending how persistent the recurrence is) EUP.

3.3.5 Feature checklists

Basically, a feature checklist is a list of an interface's features or functionality. Users are asked to tick the features that they have noticed, or used before. This information enables the investigator to see whether the features that the designer has provided are really used. Features and functionality can be listed in a number of ways, for example as a list of command names, or as semantic descriptions of particular tasks. Edgerton and Draper (1993) found that checklists offered considerable advantages over open recall, in the context of asking respondents to recall commands used at an interface.

This method is probably not such a useful source of information about the individual components of usability as some of the others, but rather a source of information about how an interface is used. However, by asking respondents for further information, say, how often they have used a feature, when the last time they used it was, and whether they know / remember how to use it, it might be possible to get an idea about the usability components. For example, if a user had never used a feature, but said they knew how to use it, it might be assumed that tasks associated with this feature would be guessable. Conversely, if a feature had been used many times, although not for quite a while, and the user had now forgotten how to use it, then the interface might have a low re-usability for associated tasks.

3.3.6 Questionnaires

These are printed lists of questions. Depending on the design of the questionnaire, the respondent may be given scales or boxes to mark, so that responses may be "scored", or they may reply to open ended questions.

The questionnaire can be a cheap method of data gathering in terms of experimenter time — having been drawn up, a questionnaire may be issued to any number of users. However, response rates for distributed questionnaires can be low (estimated at around 25% on average (Jordan and Oatley 1992), and because of this the investigator may decide to stay with respondents as they complete the questionnaire. This will, of course, radically increase the time cost. With usability questionnaires, questions generally fall into three categories: questions about how

well users think the interface is designed (e.g. "Is feedback from the interface clear?"), questions about how easy, or difficult interaction is (e.g. "Is the interface easy to use?"), and questions about feelings engendered by working with the interface (e.g. "Is working with this interface enjoyable?").

For gathering data about the components of usability, it would probably be most appropriate to ask questions about how easy, or difficult interaction was and about feelings engendered. For example, to elicit information about learnability, the question "After repeating a task a few times do you feel competent at it?" could be asked. Similarly, a question such as "Do you enjoy performing new tasks with the interface?", might yield data about the satisfaction element of guessability.

Questionnaires must be self explanatory, and it is important that they are validated before distribution. This means piloting a questionnaire, to make sure both that respondents know what they are being asked, and, in a fixed response questionnaire, that the range of responses available is appropriate. Fortunately, a number of pre-validated usability questionnaires are available (e.g. Kirakowski and Corbett 1988), including one specially designed for capturing data about the individual usability components (Jordan and O'Donnell 1992a).

3.3.7 Semi-structured interviews

With this method the investigator constructs a series of questions to use as a schedule for an interview. The investigator will then sit with the interviewee and talk informally with them around this schedule. Each item has an opening question and a number of prompts which the investigator can use to help form follow-up questions.

After the interview, the responses can be sorted into categories and quantified. However, a major benefit of this method is that the user may say something unexpected about the interface, which the investigator would not have anticipated otherwise — another potential source of useful qualitative information.

In order to use this method for the investigation of the usability components, the schedule and prompts should be adapted appropriately. For example, a schedule item may be "Were there any tasks which you found difficult at the first attempt?", which might have the associated prompt "Which ones?", or "Did you enjoy

performing task X by the time you were experienced with it?" with the prompt "Why not?".

3.3.8 Predictive techniques

A predictive technique is a non-empirical, analytic, way of estimating the usability of an interface. The intention is that neither performance data, nor subjective data from users, need be gathered. Two categories of predictive techniques are discussed here: task analyses, and "expert walkthroughs".

3.3.9 Task analyses

These techniques break down the methods for performing tasks into a series of rules — the basic idea being that the less rules required, the more efficiently a task can be completed. At their simplest, task analyses are lists of physical actions that a user must take in order to achieve a goal, however more complex analyses attempt to account for a user's mental steps as well.

Most of the models assume "expert" performance with the system. This means that the rules listed will reflect the most efficient way of performing a task, and thus may provide a useful measure of system potential, but not necessarily EUP. For example, a study by Allen and Scerbo (1983) showed that the performance of experienced subjects fell significantly below that predicted by a widely used analysis method — the keystroke model (Card, Moran, and Newell 1980) — for a text editing task.

Another type of task analysis is one which attempts to capture the concept of interface consistency. These predict how difficult a new task will be to perform, given that users are already familiar with other parts of the interface. This, then, is a form of guessability prediction (although the term "learnability" is used by those who designed the analysis methods, they do not mean the component referred to as learnability in this thesis). Probably the most widely used of this type of analysis is Task Action Grammar (TAG) (Payne and Green 1986). These analysis techniques are discussed in more detail later in this thesis, in the context of consistency related issues.

3.3.10 Expert walkthroughs

An "expert walkthrough" involves an ergonomist, or human factors specialist, analysing an interface, and then using their expertise to make judgements about its usability. These judgements shall usually be based on properties of the interface's design which might be expected to affect usability. Sometimes, these principles may be embodied in checklists (e.g. Ravden and Johnson 1989), which can allow investigators with little ergonomics training to carry out their own analysis.

Expert walkthroughs may be useful in making judgements about the quality of the overall design. However, they may be of more limited value when trying to make statements about an interface with respect to individual components of usability. This is because, whilst certain properties of an interface (e.g. screen layout, quality of error messages, visibility) are known to affect usability, it may not be known *how* they affect usability. For example, if error messages are poor, will the effects of this carry through to EUP, or will this only be important during users' early interactions?

3.4 CHOOSING A METHOD

Which method is most appropriate in any given situation will depend on the purpose of usability evaluation, and any constraints. When choosing a method it may be useful to first compare them with respect to a number of dimensions.

One dimension, which has already been mentioned, is cost. Questionnaires, incident diaries, and checklists are relatively cheap for the investigator, as it is the respondent who bears much of the cost, in terms of time, effort, and distraction from other things. At the other end of the scale are experiments, which require a great deal of preparation, and usually the presence of the investigator whilst being run.

Another dimension is the number of users likely to be needed in an investigation. Some non-experimental methods, such as think alouds and focus groups may often generate useful information with only a few subjects. However, because data from experiments is usually gathered for statistical analysis, comparatively

large numbers of subjects may be needed before anything can be reliably established.

Some of the methods are used for taking measures at the time of interaction, whist some are retrospective. With experiments performance is measured at the time, whereas with some of the non-experimental methods, such as, feature checklists or questionnaires, the user may be asked to recall previous interactions. Another issue is the type of data required. Experiments are a source of objective performance measures such as time and errors, whilst, data from some of the other methods will reflect users' subjective opinions.

The methods also differ in how much of the interface can be looked at. Experiments can be particularly useful for focussing on specific design issues. For example, if the effect of a particular design property on usability were being investigated, this property could be varied between tasks or between interfaces, to allow comparisons of performance. At the other end of the scale, expert walkthroughs can be an efficient method for producing general comments about the interface as a whole.

The experiment was the method used in the studies reported in the next part of this thesis. The studies were designed to investigate the effect of manipulating a specific design property — consistency — and the intention was to test particular theories. It was, therefore, necessary to have a source of fairly unambiguous data. The dimensions on which experiments can seem unattractive — cost to the investigator, number of subjects required, and narrowness of focus — would not, it was anticipated, cause too many problems in this context.

3.5 SUMMARY

An experimental methodology for quantifying the various components of usability has been proposed, and a range of possible analysis techniques considered — including criteria based analysis. Alternative, non-experimental, methods have also been reviewed, with suggestions as to how they may be used with the multi-component usability framework. Finally, the methods have been compared on a number of dimensions.

Chapter 4

APPLYING THE FRAMEWORK TO A USABILITY EVALUATION OF A VIDEO CASSETTE RECORDER

This chapter reports on an investigation of the usability of a video cassette recorder (VCR). The study had two main aims. The first was to provide an illustration of how an experimental technique coupled with a criteria based analysis could be used to quantify the five usability components. The second was to show that, by applying the multi-component framework, it was possible to identify bugs which affected the various components of usability in different ways.

4.1 THE STUDY

4.1.1 The VCR

The VCR used for the study was a Sharp VC-A140HM, probably a mid-range model, with most of the features usually associated with VCRs, including a timer facility which could be programmed for recording. The VCR's control panel is illustrated in figure 4.1.

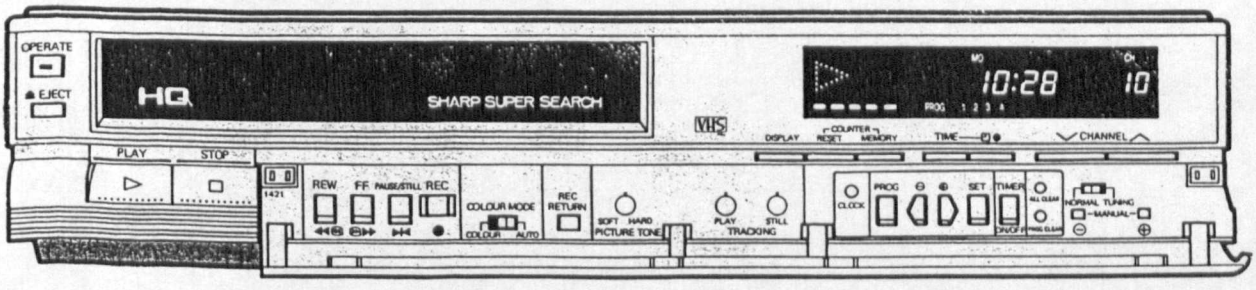


Figure 4.1. VCR's control panel.

4.1.2 Subjects

Twenty-eight female students participated in the study. All had previous experience of VCR use, but none had used the experimental VCR before, nor one made by the same company.

4.1.3 Experimental design

The subjects were treated as a single group. Each subject was to be set tasks over two experimental sessions. The first session included ten trials, each containing one experimental task, consisting of six sub-tasks. These sub-tasks — listed in table 4.1 — were chosen to represent the range of the VCR's functionality. The contents of each trial were treated as a single task, as sub-tasks were always presented in the same order, giving no balance for order effects. Therefore results would not necessarily reflect the usability of each sub-task when considered in isolation. In the second session, which was approximately 24 hours after the first, subjects were asked to complete a further trial.

1. Set the clock to a specified time and day.
2. Insert a pre-recorded tape, and re-wind it to the beginning.
3. Fast forward the video to a specified location (defined by counter position) and play.
4. Stop the tape and change the TV channel.
5. Eject the pre-recorded tape, and replace it with a blank one.
6. Program the timer to record from a specified channel at a specified time.

Table 4.1. Sub-tasks set to subjects.

If subjects got stuck when performing any of the sub-tasks they could ask the experimenter for help. Subjects were told that getting stuck meant not knowing what to do in order to be able to progress on a sub-task for more than about two minutes. The experimenter would then intervene at one of four levels, depending on how far the subject had already progressed. If a subject had altered the state of VCR and could not get back, the experimenter returned the VCR to its state before the sub-task was attempted. If a subject didn't know what to do and had not yet consulted the manual, which was freely available throughout the experiment, then the experimenter would simply advise her to do this. If the manual, had been consulted but the subject had not found the appropriate instructions, then the

experimenter would open the manual at the appropriate page. Finally, if the subject had read the appropriate instructions but still could not perform the sub-task, then the experimenter would "talk her through" step by step.

The measures of performance taken were: time on task, number of errors made, number of sub-tasks per trail for which manual was consulted, and number of interventions made by the experimenter.

4.1.4 Procedure

After reading an introduction to the experiment (appendix 4.1), subjects were issued with a sheet listing the sub-tasks they were to complete for the first trial (appendix 4.2). They were told to read this through, to ensure that they were clear about what they were being asked to do. The experimenter answered any queries related to clarifying what subjects had been set, but did not answer questions about how sub-tasks could be completed.

Subjects were told that they could either spend some time in free exploration of the machine, or start the first experimental trial straight away. The reason for this choice being offered, was in order to introduce an element of "ecological validity" to the study — some users may prefer to explore a new machine before attempting any specific task in order to get a general overview of the VCR. If any subjects did decide to explore the VCR first, then the time they took was counted as part of the first trial time (only two of the twenty-eight subjects did this). It may appear that this opportunity for free exploration amounts to introducing an arbitrary degree of practice — an approach criticised in previous chapters. The difference is that here the subject already knows what the goal is, and the time taken to achieve that goal for the first time was recorded. By allowing free exploration, the experimenter was merely giving the subject the option of choosing her preferred approach to achieving that goal.

As soon as a trial was completed, the next one was set straight away. The sub-tasks differed from trial to trial in that the clock and timer settings, the section of the video cassette to be played (the pre-recorded cassette was a collection of pop videos, so the sections were identified by the pieces of music with they contained), and the TV channel to be selected were all varied. As the subjects worked through the trials, the experimenter made a record of any errors that were

made, and noted the performance measures. After ten trials had been completed the first experimental session, which took about an hour, was over.

About 24 hours later, the subjects returned for a second experimental session. This involved completing one further trial. The experimenter followed a written protocol throughout both sessions (appendix 4.3), in order that subjects were treated in a fairly uniform manner.

4.2 RESULTS, ANALYSIS AND DISCUSSION

Figures 4.2, 4.3, 4.4, and 4.5, illustrate mean performance against trial number in the first session, in terms of time on task, errors, manual consultations, and experimenter interventions respectively.

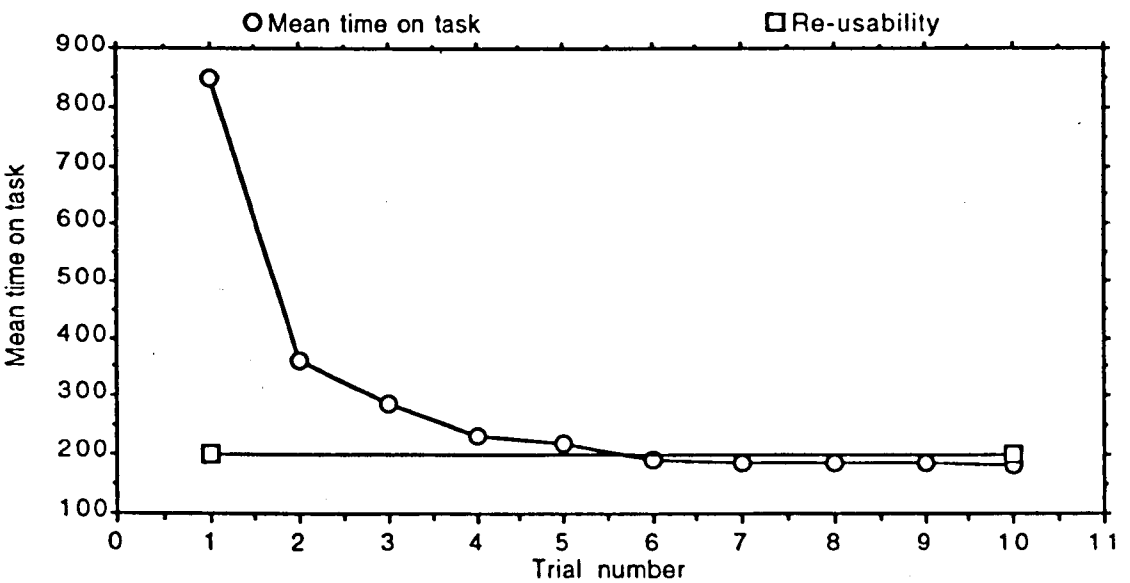


Figure 4.2. Mean time on task (seconds) per trial.

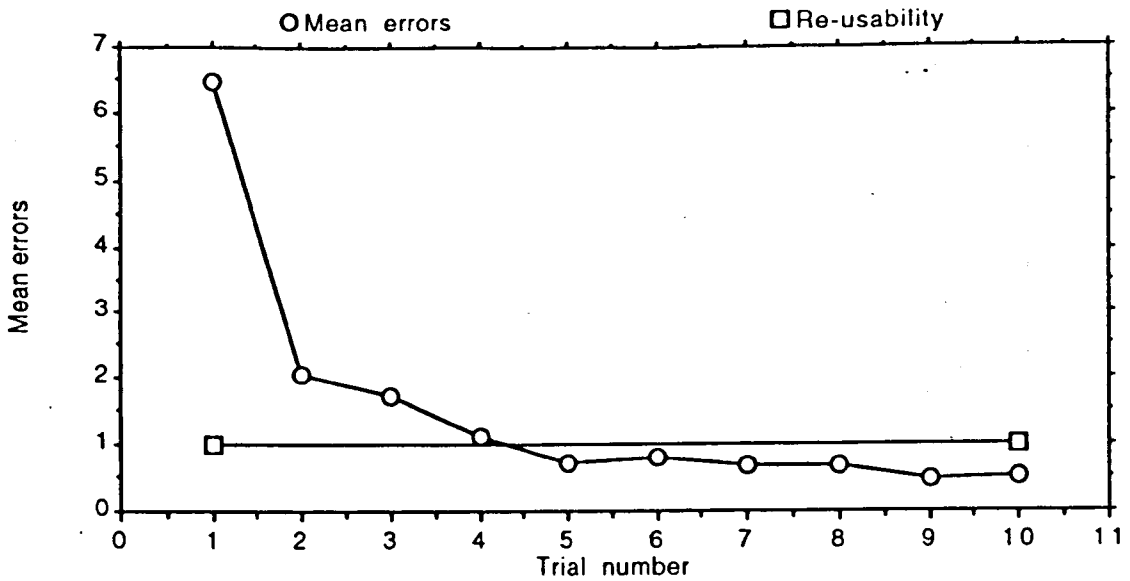


Figure 4.3. Mean number of errors per trial.

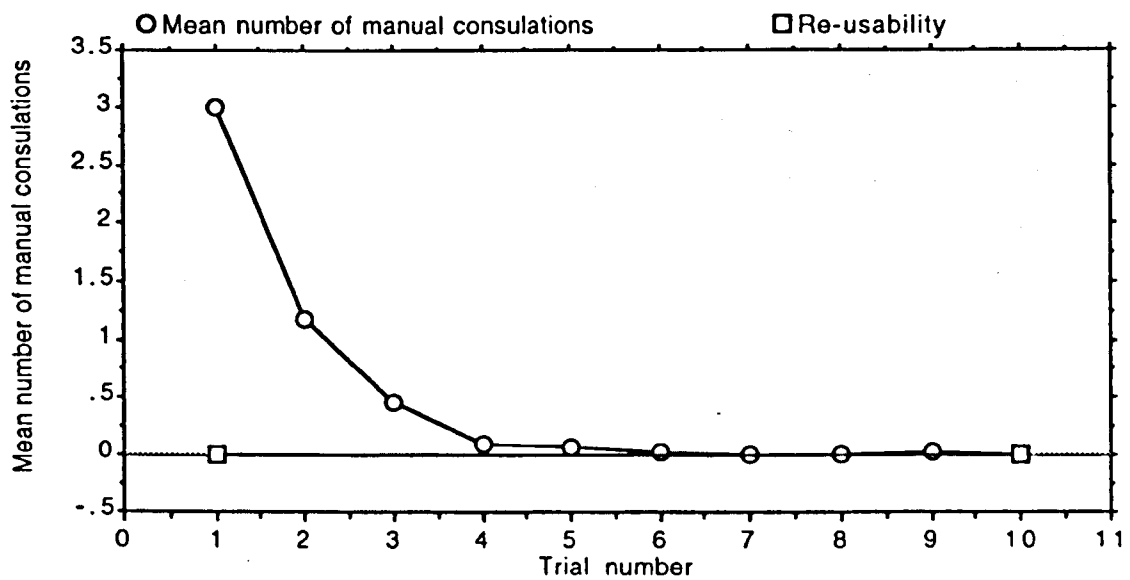


Figure 4.4. Mean number of sub-tasks for which manual was consulted per trial.

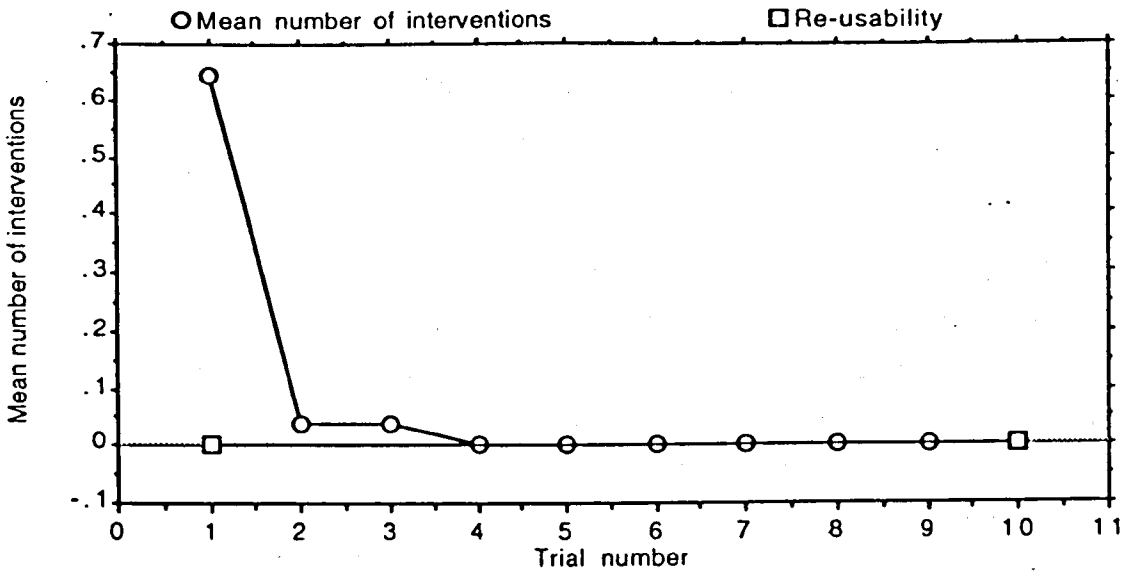


Figure 4.5. Mean number of experimenter interventions per trial.

In the previous chapter a criteria based analysis for quantifying the various components of usability was outlined. Where comparisons between different interfaces or different tasks are to be made, or the components of usability are to be evaluated in relation to a priori benchmarks, such quantification will be necessary. This is the case in the studies reported in the next part of the thesis, where performance on different types of task is compared. This type of analysis is performed here, mainly for demonstration purposes. However, in the context of this study, perhaps the main benefit of evaluating the VCR was in identifying bugs which affected the various components of usability. Following the criteria based analysis, there is a discussion of the bugs found, demonstrating how the multi-component usability framework aided in their identification and classification.

4.2.1 Sensitivity of performance measures

Note that the curves all appear to contain an area where performance levels off, but where this occurs appears to depend on the measure of performance being used. For example, the number of experimenter interventions drops away very markedly after the first trial. By trial two the experimenter is making about eighteen times fewer interventions than on the first trial, and by trial four, the rate of interventions has fallen away to zero.

The rate at which the other measures head toward their lowest values is slower. For example, errors diminish by about two thirds, between trials one and two, and

appear to stabilise around trial five. Time on tasks drops even less rapidly, by just over a half from trials one to two, with a stable level of performance starting around about trial seven.

When choosing measures of performance, it may be worth bearing in mind their comparative sensitivities. For example, the time and error data clearly indicate that subjects' performance continues to improve after trial four. However, had performance been measured purely in experimenter interventions, this would not have been picked up. So, although subjects no longer needed experimenter help after trial three, mean performance was still improving at least up to trial five. Similarly, if interventions and manual consultations had been the only measures used, it might be concluded that subjects were having no problems by the end of the first experimental session. The error data, though, indicates that some were. It would appear, then, that of the measures employed here, time on task is the most sensitive. There could, however, be even more sensitive measures than this, which would perhaps continue to show improvements with experience even after time on task had stabilised. A possibility is mental workload (Jordan 1990 describes mental workload and techniques for measuring it).

4.2.2 Guessability

Values for guessability were taken from performance on the first trial of the first experimental session. This gave the mean values listed in table 4.2.

Time on task	Errors	Manual consultations	Experimenter interventions
848 s (sd = 402 s)	6.46 (sd = 3.82)	3.00 (sd = 1.09)	0.64 (sd = 0.91)

Table 4.2. Mean guessability values.

4.2.3 Learnability

A criterion based approach was taken to decide when a subject could be said to have "reached a competent level of performance on the experimental task". It was decided that, in terms of the various performance measures, a subject had learned the task if their performance satisfied the criteria listed in table 4.3

Time on task	Two consecutive trials, each completed in less than four minutes (240 s)
Errors	Two consecutive error free trials.
Manual consultations	No longer needing to consult the manual.
Experimenter interventions	No longer requiring experimenter interventions.

Table 4.3. Criteria set to represent learning the experimental task.

The criteria relating to time on task and errors, both include the stipulation that performance should be maintained over two consecutive trials. This was done for two reasons. Firstly to avoid the results being distorted by one-off "lucky strikes". It might be, for example, that a user managed to "muddle through" in less than four minutes during, say, her second trial, but then went on to perform more slowly on the next few trials. It would not appear, then, that taking this subject as having learned the interface after two trials would really be representative of her degree of competence at that stage. Secondly, the "two trials" criteria can also allow for slips. Had subjects been required to maintain the set level of performance until the end of the session a one-off drop in performance might cause a distortion of the data. For example, if due, say, to a lack of concentration, a subject, whose performance had been error free for several previous trials, made an error, performance on this trial would not necessarily reflect her level of competence. The relatively complex nature of the experimental task made such a scenario a possibility. Performance in the first trial was not considered in this analysis (it had already been considered in the guessability analysis). Therefore, learnability values were between 1 and 9 trials. (So, for example, if a subject fulfilled a criterion, in terms of one of the experimental measures, on, say, the fourth trial of the session, she would be given a learnability value of three trials with respect to that particular measure).

If a subject's performance did not fulfil the learnability criteria with respect to any of these measures, a default value of ten trials was allocated in terms of this measure. This implicitly assumed that the criterion would be met had the subject continued for another trial. Whilst there is no reason to expect this assumption to be correct, this approach means that results give a lower bound of the mean number of trials taken to learn the interface. Three performances failed to meet the criteria in terms of time, and one in terms of errors.

The levels in the criteria were set post-hoc. This was done in accordance with the aims of the study — to illustrate how the components of usability might be quantified. For example, most subjects managed two consecutive trials in less than four minutes each by the end of the first session, hence this was thought to be a sensible time criterion. In a design context however, these levels could be set beforehand as part of an interface's usability specifications. For example, it might be decided that, say, 90% of users should be able to complete the experimental task in less than five minutes, by their fifth trial.

In accordance with the criteria set, the mean learnability values for this interface are listed in table 4.4.

Time on task	**Errors**	**Manual consultations**	**Experimenter interventions**
4.0 (sd = 2.6)	2.9 (sd = 2.1)	2.5 (sd = 1.9)	1.1 (sd = 0.6)

Table 4.4. Mean trials to fulfilment of learning criteria.

4.2.4 EUP

Measures of EUP were taken from subjects' mean level of performance over the last three trials of the first session. Implicitly, this means that subjects were defined as being "experienced" after having completed seven trials (i.e. before embarking on their eighth). Again, the reason for choosing to take measures over a three trial run, rather than, say, merely using data from the last trial, was to minimise noise from one-off performance fluctuations. This would include slips but may also include "end of session highs" which can cause uncharacteristic surges in performance at the the end of an experimental session.

The values obtained for EUP are listed in table 4.5

Time on task	**Errors**	**Manual consultations**	**Experimenter interventions**
186 s (sd = 41 s)	0.57 (sd = 54)	0.01 (sd = 0.19)	0.00 (sd = 0.00)

Table 4.5. Mean EUP values.

4.2.5 System potential

Potentially a user could achieve error free performance, without consulting the manual, or requiring any interventions from the experimenter. However, in terms of time on task, the system potential is less obvious. The approach taken here was to break down the methods for performing each sub-task into a series of action rules. The action-rules chosen represented what the experimenter thought were the most efficient methods of performing each sub-task, and coincided with those given in the manual (see appendix 4.4). Times were then allocated to each action on the basis of how quickly they could be performed by an "over-practiced expert". In this case, the experimenter — who had extensive practice with the VCR before designing the experiment, and had watched twenty-eight subjects complete the experimental task eleven times each — was taken as being expert.

The experimenter performed each stage of each sub-task five times, the quickest of these being used in the analysis. Overall, the value calculated for system potential was 96 seconds. System potential values are summarised in table 4.6. Sub-tasks were performed separately so that any slips made by the experimenter wouldn't necessitate a repetition of the entire task.

Although system potential is theoretically a property of an interface alone, this analysis, in terms of time on task, was heavily dependent on the experimenter. The most efficient method of task performance was left to his judgement, and the time to complete the task constructed from his performance on the sub-tasks. Although it may seem regrettable that an individual's performance is being used to quantify something which is an interface property, there does not seem an obvious practical way around this. Even if a formal method such as a task analysis were used, judgements about the most efficient methods of task performance would still be required, and there would also be the issue of assigning times to each action or thought. Perhaps the most constructive way of approaching this problem is simply to ensure that the "expert" used is someone appropriate in the light of the judgements and performance demanded of them.

Time on task	Errors	Manual consultations	Experimenter interventions
96 s	0	0	0

Table 4.6. System potential values.

The gap between mean EUP and system potential in terms of time seems, perhaps, surprisingly large. However, this may be partly reflecting the experimenter's remit to complete the task as quickly as possible. This meant, for example, pressing and moving between buttons, as fast was physically possible for the experimenter. Subjects may not have felt any incentive to do this — certainly, they appeared to be working well within their physical and motor limitations.

4.2.6 Re-usability

Performance on the trial in the second session was taken as a measure of re-usability, giving the mean values listed in table 4.7.

Time on task	Errors	Manual consultations	Experimenter interventions
201 s (sd = 27 s)	1.00 (sd = 0.94)	0.00 (sd = 0.00)	0.00 (sd = 0.00)

Table 4.7. Mean re-usability values.

Performance is at the same level as EUP in terms of manual consultations and experimenter interventions (in fact marginally better in terms of manual consultations, as one subject consulted the manual once during the last three trials of the first session). However, in terms of time on task and errors performance has slipped back to approximately the levels of trials 6 and 4 in the first session respectively.

Of the components of usability measured, the values obtained for re-usability are perhaps the most difficult to interpret. Clearly, it might be expected that the length of time away from the interface would affect performance. So, had subjects returned after a week, rather than a day, performance might have dropped further. Also, the shape of a user's learning curve may be different for the returning user. Here, for example, the mean re-usability value in terms of errors is equivalent to performance at about trial 4 in the first session. However, this does not necessarily mean that if subjects were to do the task again their performance would be equivalent to that in trial 5, and so on. Rather, it could be, say, that performance gets back to EUP after only another couple of trials, perhaps due to the user's memory being "jogged", by the first re-usability trial.

4.3 IDENTIFYING AND CLASSIFYING BUGS

So far a series of values have been obtained, giving ratings of the various components of the VCR's usability with respect to an experimental task. As discussed earlier, had the study been conducted for comparison purposes, or to evaluate usability against benchmarks, these results should prove both meaningful and useful. However, as there were no set goals or comparisons to be made in this study, these results are not particularly meaningful in their own right. Nevertheless, the experimenter's notes were still useful in identifying bugs. In this section the bugs which caused the most problems are identified and classified according to which components of usability they affected. Possible design solutions to alleviate these problems are then suggested.

On the first trial there were four types of error that were made by ten or more of the subjects: pressing the "TIME" button rather than the "CLOCK" button in sub-task 1 (11 subjects), invoking picture search in sub-task 2 (18 subjects), stopping the tape in the wrong place in sub-task 3 after fast-forwarding (16 subjects) (subjects were expected to stop the tape within five counter numbers of the ideal place), and exiting from programming mode whilst setting the timer to record in sub-task 6 (17 subjects). So, these problems might be indicative of the VCR's main guessability bugs.

When subjects started the first trial they were presented with the VCR with its front panel closed. When asked to set the clock, it is, perhaps, not surprising that the button which appeared most obvious to many was the one marked "TIME" under the display. A possible solution to the problem might be to have clear markings on the front panel so that users know it can be opened (some appeared not to realise this). A bigger "CLOCK" button might also have been helpful — not only was the small button not visually salient, it also proved difficult to press for subjects with long fingernails. Apart from this, problems with clock setting were not particularly persistent for the majority of subjects. By trial two only two subjects repeated their errors, and by trial five the problem had gone.

Unintentionally invoking the "Picture Search" facility during sub-task two was a far more persistent problem. In fact more subjects (nineteen) made this error on trials two and three than on the first, eight were still invoking it by the end of the first session, and eleven on their re-usability trial. The problem here was caused by an automatic play facility, which meant that any pre-recorded tape would be

played automatically as soon as it was inserted into the machine. This meant that pressing the re-wind button would not cause the tape simply to be re-wound, as it would had the tape been stopped, but rather that the machine would picture search through the tape. This meant that instead of the start of the tape being quickly reached, the contents of the display were shown as the tape re-wound, slowing it down considerably.

At least two aspects of ergonomic design had been violated here. Firstly, control had been taken away from the user. If users had wanted to play the tape they could press the "PLAY" button. Presumably the automatic play facility was introduced for convenience, but in fact many tasks are made longer by this facility as the tape must be stopped before, for example, re-wind, fast forward or record can be invoked. Secondly, the design was inconsistent with other aspects of the machine. None of the other facilities are invoked automatically, so why should this one be? Many of the subjects did not appear to realise that there was an automatic play facility, but rather were under the impression that whenever the tape was re-wound its contents were displayed automatically. It is not surprising, then, that this proved a persistent problem. The solution? — remove the automatic play facility.

Stopping the tape in the wrong place on sub-task 3 was another fairly persistent problem, Eight and twelve subjects had this problem on trials two and three respectively, by the end of the session this was still causing problems (e.g. for six subjects on trial eight, and four on trial ten), as in the re-usability trial (six subjects). The problem was that the speed at which the tape forwarded, and hence the rate at which the counter display changed, made it difficult for subjects to react in time. Although this was a persistent problem it was also a fairly minor one in terms of its associated time penalty — subjects could clearly see what the problem was and knew how to rectify it. However, if a solution were sought, the option of pre-programming a counter setting at which to stop the tape would be a possibility.

When setting the timer to record (sub-task 6), the VCR automatically exits from programming mode if the user has not pressed a button for over a minute. This was the major source of guessability problems in sub-task 6. Again, the fault here is that control is being taken away from the user. When a subject was working their way through the task for the first time they would often read the manual as they progressed, taking the task stage by stage. This strategy meant that there

might often be a long gap between button presses, and that, to the bewilderment of the subject, programming mode had been exited by the time they attempted the next part of the task. Again, the solution appears to be to remove the automatic mode exiting feature. This problem became less significant as subjects gained experience, as they were no longer taking over a minute to make an entry. By the second trial only two subjects had problems with this, and the problem didn't arise after trial four.

Of the four major guessability problems two appeared to persist throughout the first and second experimental sessions, whilst the others disappeared relatively quickly. It might be said, therefore, that as well as affecting guessability, two of the problems also affected learnability, EUP, and re-usability, whist the other two had a slight effect on learnability.

There were other problems which, whilst not perhaps affecting guessability as much as some others, had a comparatively bigger effect during subsequent trials, thus affecting learnability. The two most noticeable both occurred on sub-task six.

One of these was that subjects attempted to use the channel changing buttons when programming the number of the channel to be recorded. This is a type of mode error — in television viewing mode the channel changing buttons are used to alter the channel setting, whilst in programming mode the "+" and "-" programming buttons are needed. The solution might be to allow the changer buttons to be active in either mode (whilst still retaining the option to use "+" and "-" buttons in programming mode, so as not to risk interference from other programming tasks). Only three subjects had problems with this on the first trial, making it relatively negligible as a guessability problem. However, six and four subjects had problems on trials two and three respectively, and three were still having problems by trial six, making it the joint second most common problem at this stage (behind invoking picture search, and level with stopping the tape in the wrong place when fast forwarding). The problem had disappeared by the final three trials, but re-occurred in the second session. It might be classified, then, as affecting learnability and re-usability, and having a slight affect on guessability.

The second problem was that subjects sometimes entered the wrong settings when programming the timer. For example, a subject might set the timer to record on, say, Monday, rather than Sunday. A common cause of such errors was keeping the "SET" button held down for too long after making an entry. This had an effect

equivalent to pressing the button twice, not only setting the intended parameter, but also confirming the default setting for the next parameter. When this occurred, subjects had to keep pressing "SET" until they were out of programming mode (having programmed in all the default settings). They then had to re-enter the programming mode in order to put in the appropriate settings. This meant paying a comparatively large time penalty for a fairly trivial error. A way round this problem is to allow users to move backwards and forwards through the programming sequence, so that, for example, the user could set the day after the starting time. This principle is referred to by Thimbleby (1991) as making the machine free of history. This problem occurred eight times during the first trial, and seven during the second, making it the third most common problem on this trial, it was still occurring by the end of the first session, and in the second session. Thus the problem affected guessability, learnability, EUP, and re-usability.

By identifying how various problems affect the different components of usability, it is possible both to classify the effect of any particular bug, and to set an agenda for design changes. For example, it might be decided that high learnability and EUP were desirable, but that the level of guessability didn't matter very much. If this were the case, the designer may decide to concentrate on alleviating the problems that persisted throughout the session (e.g. the unintentional invocation of picture searching and the mode error problems associated with the channel buttons), rather than those which only caused problems at the start (e.g. finding the "CLOCK" button and the time out feature of the programming facility). Conversely, if guessability were regarded as the priority, these latter changes would be amongst the most important.

The usability framework, then, has provided a useful way of classifying problems. Furthermore, it is doubtful that such a range of bugs would have been discovered had a performance profile not been built up over several trials. Consider, for example, the guessability problems associated with finding the clock setting button. Had subjects been given a practice session followed by a one off trial, this problem may have disappeared, and the bug gone unnoticed. Similarly, if subjects were set a one off task, with no practice, then the problems which only affect small numbers of users yet are persistent might not be thought significant (as there would be no way of telling that they were persistent). Here, this would include the problems of users trying to program the channel to be recorded using the channel changer buttons.

4.4 SUMMARY

A study has been reported, illustrating the application of an experimental methodology and a criteria based analysis to quantify the components of usability, in the context of evaluating a VCR. Values for each usability component have been obtained in terms of four measures — time on task, errors, manual consultations, and experimenter interventions. In other contexts these could be used for, say, comparison between interfaces, or checking whether an interface met specified standards.

Employing a multi-trial methodology revealed a range of bugs, many of which may not have come to light in a single trial experiment, and the multi-component usability framework proved useful in classifying the effects of each.

PART 2: CONSISTENCY

Chapter 5

WHAT IS CONSISTENCY?

One of the most widely recognised principles of designing for usability is the desirability of consistent interfaces (e.g. Ravden and Johnson 1989, Shneiderman 1987). The comment from Smith and Mosier 1986 that "... the single objective upon which experts agree is design consistency", is probably only a mild exaggeration. However, formal definition of consistency has proved difficult. Grudin (1989) claims that in 1988 a two day workshop containing fifteen experts was unable to arrive at a definition. He goes on to compare the designer striving for consistency to the supreme court justice trying to define pornography "... each of us feels we know it when we see it, but people often disagree and a precise definition remains elusive."

This chapter contains a review of previous work on consistency — describing how understanding of the issue has developed with successive treatments. The concepts are then taken a stage further, by introducing a new distinction. Two types of consistency — set compatibility and rule compatibility — being identified and discussed. Definitions of these are then offered.

5.1 TOWARDS UNDERSTANDING CONSISTENCY: THE DEVELOPMENT OF FORMALISMS.

A useful starting point is to think of consistency loosely as "doing similar things in similar ways" (Reisner 1990). The next step then is to try and define what is meant by "similar things" and "similar ways". Several attempts at doing this have been made via the development of formalisms to identify consistency (and inconsistency) (Reisner 1981, Payne and Green 1983, Kieras and Polson 1985, Payne 1985, Payne 1985a, Payne and Green 1986, Payne and Green 1990, Reisner 1990).

A common theme of these formalisms is to break down methods for performing tasks into a series of action-rules. In this context defining similar things means deciding when two or more tasks are similar, whilst defining similar ways means deciding when two or more action-rules are similar. If a set of similar tasks could

all be performed using a set of similar action rules, then an interface would be considered consistent for those tasks. However, if one or more of the set required a different type of rule, the interface would be considered inconsistent.

Similarities between action rules are captured via higher level rules called rule schemata. These are generalised rules, into which variables can be substituted to represent individual cases. If the rules for performing all the tasks in a particular set (called a schema set — the set of tasks upon which a user would expect a single schema to operate) could be represented by a single schema then the interface would be considered consistent for that set of tasks. However, if two or more schemata are required per schema set, then there is an inconsistency. The concepts of rule schemata and schema sets will be illustrated by examples in the following sections of this chapter.

Although there is a consensus that inconsistency within a set of task-action rules can be spotted by the extra rule, it is less clear how a "schema set" — the group of similar tasks upon which a user would expect a single schema to operate — is defined. This is where the major differences between the formalisms lie — the criteria for grouping tasks into schema sets becoming increasingly explicit with successive treatments. In the following sections a brief outline of the principles underpinning each formalism is given, and illustrated by example. For the sake of clarity and uniformity, the exact notations may differ slightly from those originally used.

5.1.1 Reisner's Action Language

This was perhaps one of the earliest attempts to apply a formal grammar in the human factors area. Reisner (1981) applied a notation similar to Backus-Naur Form (BNF) to describe the "action language" of two different graphics programs, called ROBART 1, and ROBART 2. In ROBART 1 the methods for selecting lines, circles, and squares were all similar, the user having to first set an appropriate switch. However, the method for selecting text was different, users being expected merely to start typing.

With ROBART 2, the methods for selecting lines, circles, and squares were compatible with the method for selecting text — the user having to move a cursor

over an appropriate icon. Reisner represented the rules for the programs as follows.

ROBART 1
to select line -> set "line" switch + go
to select circle -> set "circle" switch + go
to select square -> set "square" switch + go
to select text -> go

ROBART 2
to select line -> move cursor over "line" icon + go
to select circle -> move cursor over "circle" icon + go
to select square -> move cursor over "square" icon + go
to select text -> move cursor over "text" icon + go

Table 5.1. Reisner's action language descriptions of ROBART 1 and 2.

Although the overall number of rules associated with each program is the same, it is possible to combine the rules associated with ROBART 2 into a single general rule schema (which she termed a "necessary" rule), whereas this cannot be done with ROBART 1 (table 5.2), where a separate rule schema is needed to deal with selecting text. There are further rules to define which items can be used in a rule schema.

ROBART 1
Rule schemata
to select SHAPE -> select SHAPE switch + go
to select TEXT -> go

SHAPE -> line | circle | square
TEXT -> text

ROBART 2
Rule schema
to select SHAPE -> move cursor over shape icon + go

SHAPE -> line | circle | square | text

Table 5.2. Reisner's "necessary" rules (rule schemata) for ROBART 1 and 2.

In this simple example the inconsistency in ROBART 1 could be identified by the additional rule schema. As this rule was associated with selecting text, it was here that Reisner predicted that users would have difficulties. This proved to be the case, with users spending time looking for a text switch that did not exist. No comparable errors occurred with ROBART 2.

In this treatment the idea of identifying inconsistency by an extra rule was introduced. This addresses the question of whether tasks are performed in "similar

ways". However, the question of what are "similar things" has not yet been addressed. Presumably, Reisner is taking all tasks which can be performed using the interface as being similar.

5.1.2 Production rules

Production rule analysis was developed by Kieras and Polson (1985), as part of their cognitive complexity theory (CCT). Each rule takes the form of IF (condition) THEN (action), and a set of one or more of these will be required in order to perform any given task. The idea is that the difficulty of performing new tasks can be estimated from the number of new rules required. A new rule is defined as being one which is unique to a particular task, or one which is being encountered for the first time. Therefore, both consistency and order effects are addressed. A basic consistency metric might be the number of common production rules in a set of tasks — the more rules which are common between tasks, the more consistent an interface is. Kieras and Polson predict a linear relationship between the number of new rules and the time it takes a user to complete a new task.

Broad empirical support has been found for the theory (e.g. Polson 1988, Lee, Polson and Bailey 1989), although the results of other studies (Karat, Fowler and Gravelle 1987, Vossen, Sitter and Ziegler 1987), suggest that a treatment which takes no account of the comparative complexity of different rules might sometimes be too simplistic.

Like Reisner (1981), Kieras and Polson do not explicitly address the issue of similarity between tasks.

5.1.3 Set Grammar

This formalism was developed by Payne and Green (1983). It represented a progression over Reisner's notation, as the notion of sets ("similar things") was made explicit. Indeed the set — a grouping of tasks that had something in common — formed a starting point for this notation.

Payne and Green regarded the sets as being representative of the way that the user would group tasks, rather than as a function of the interface — they called this "grammar in the head". With this formalism, the analyst must use his or her judgement to predict users' sets. In the example below (table 5.3), the analyst has put all items which may be selected into a single set called "SHAPES". Note, however, that a set's name is merely a label, and is not, therefore, central to the analysis. It would make no difference if the set were labelled, say, "ITEM", or "UNIT".

Again, the inconsistency in ROBART 1 can be identified from the extra rule schema. However, the reason why the schema is considered extra is now explicit. It is extra because it is associated with selection of one of the items in the set labelled "SHAPES". Had the analyst regarded selecting text as being in a separate set from selecting lines, circles, and squares, then there would not have been any inconsistency.

Table 5.3 gives a Set Grammar notation for ROBART 1 and 2:

ROBART 1
Schema set
SHAPES: [line, circle, square, text]

Rule schemata
select SHAPE: -> set SHAPE switch + go (for shape = line, circle, square)
select SHAPE -> go (for SHAPE = text)

ROBART 2
Schema set
SHAPES: [line, circle, square, text]

Rule schema
to select SHAPE: -> put cursor in SHAPE icon + go

Table 5.3. Set Grammar notation for ROBART 1 and 2.

The central point here, then, is that set grammars support the distinction between the range of tasks which can be performed on an interface, and those which users might expect to be performed in similar ways (using the same rule schema).

In the example above, the analyst has put all four tasks into the same set and so, as with Reisner's (1981) notation, the analysis indicates the presence of an inconsistency. However, consider the alternative analysis for ROBART 1 below (table 5.4). Here the analyst has made a distinction between items which he or she has labelled "SHAPE", and those labelled "TEXT". The analyst, then, has

indicated that he or she does not think that users would expect selecting text to be performed in a similar manner to selecting lines, circles, or squares. Thus the overall set of tasks has been divided into two separate schema sets. This analysis indicates that the interface is consistent, as although there are a total of two rule schemata, there is only one per schema set.

ROBART 1
Schema sets
SHAPES: [line, circle, square]
TEXT: [text]
Rule schemata
select SHAPE -> set SHAPE switch + go
select TEXT -> go

Table 5.4. Alternative Set Grammar notation for ROBART 1.

5.1.4 Task Action Grammar (TAG)

TAG (Payne 1985, Payne and Green 1986, 1990) is based on the same principles as Set Grammar — grouping tasks, and looking for "family resemblances" between rules. However, the "something in common" by which items are grouped together, have now been identified as common semantic features. Table 5.5 gives a TAG description of the ROBART interfaces. Note, that the "task sets" notation refers to any tasks which could be performed using the interface. Schema sets are identified from the semantic feature in the square brackets. Here, all tasks are in the same schema set, as — in the opinion of the analyst — all objects share a common semantic feature "shape".

ROBART 1
Task set
select line
  [shape = line]
select circle
  [shape = circle]
select square
  [shape = square]
select text
  [shape = text]

Rule schemata
select (shape = ONE OF: [circle, square, line]) -> set switch (shape) + go
select (shape = text) -> go

ROBART 2
Task set
select line
  [shape = line]
select circle
  [shape = circle]
select square
  [shape = square]
select text
  [shape = text]

Rule schema
select shape -> put cursor in shape icon + go

Table 5.5. TAG description of ROBART 1 and 2.

The reason for the second schema associated with ROBART 1 being classified as "extra" is that (according to this analysis) it contains the same semantic feature as the first (here it is labelled "shape" but it could equally well be labelled, say, "unit" (Hoppe , Tauber, and Ziegler 1986) or "things to be selected" (Reisner 1990)).

TAG has become a widely referred to formalism. Variants on the grammar have also been created to adapt it to new types of interface, in addition to the command driven interfaces for which it was originally intended — for example Howes and Payne's (1990) D-TAG — a formal notation for display based competence.

5.1.5 Agent Partitioning Theory (APT)

Reisner (1990), whilst recognising the need to establish common criteria for putting tasks into sets, points out that, unfortunately, there are not universal laws of semantic grouping. As she says, if there were we could simply "...look for them (semantic groups), hope to eventually come up with a definitive set, analyse

tasks into semantic features, and automatically identify inconsistency." (This is what TAG aims to do). Rather, she claims, it is more constructive to simply think in terms of users giving labels to things, and putting tasks into sets on the basis of them sharing a common label. Based on this idea she has developed a framework for identifying (in)consistency — Agent Partitioning Theory (APT).

The basis of APT is that semantic groupings are simply decided by individual people (agents). Inconsistencies, then, may be viewed as arising when a user's grouping's are different from those embodied in the interface (which represent the designer's groupings). So inconsistency is a property of a human-machine interaction — dependent both on the individual user and the design of the interface — rather than being solely a property of either the interface, or grammar in the head. This is the key point made by Reisner (1990) that had been missed by earlier work.

To capture inconsistencies, then, it is necessary to know, firstly which tasks belong in which set, and secondly who says so. To describe which things belong in which schema set, assignment rules are required, for example, circle => [SHAPE]. To describe who says so it is necessary to specify an assigning agent, this may be the designer or the user of a machine. Reisner calls the combination of the assignment rule and the assignment agent a partitioning rule; an example might be: circle => [SHAPE] for agent designer. The central claim of APT is that when a system is inconsistent different assigning agents use different assignment rules. Specifically, in the case of interface inconsistency, the designer (whose assignments are embodied in the machine) and the competent user apply different assignment rules. A "competent user" is one who makes mistakes common to many users, and probably ascribable to some system design feature; however, the competent user does not make spurious errors. So in a consistent system both user and designer will partition a given overall task set in the same way (they will divide it into similar schema sets). These schema sets can then be used in the generalised rule schemata — the similar ways of handling similar things.

Returning to the original example, consider two different users (User A and User B) using ROBART 1. User A regards selecting lines, circles, squares, and text as being similar tasks (perhaps he or she thinks of them all as "selecting tasks"). User A will find ROBART 1 inconsistent, as all four tasks cannot be performed similarly using the interface. However, User B regards selecting lines, circles, and squares as being similar tasks (perhaps as "selecting geometric shapes"), but

thinks of selecting text as being different. User B will find ROBART 1 consistent — as for this person there will only be one rule schema per set. A system, then, will be consistent if the user has partitioned tasks into sets in the same way as the designer, but inconsistent if the user and designer have partitioned tasks differently. Table 5.6 summarises ROBART 1 for users A and B, using the APT notation.

Task set
TASKSET = [select line, select circle, select square, select text]

Schema sets (agent = designer)
SHAPE = [line, circle, square]
TEXT = [text]

Rule schemata (agent = designer)
to select SHAPE -> set SHAPE switch up + go
to select TEXT -> go

Schema sets (agent = User A)
SHAPE = [line, circle, square, text]

Rule schemata (agent = User A)
to select SHAPE -> set SHAPE switch up + go
(SHAPE -> line | circle | square)
to select SHAPE -> go
(SHAPE -> text)

Schema set (agent = User B)
SHAPE = [line, circle, square]
TEXT = [text]

Rule schemata (agent = User B)
to select SHAPE -> set SHAPE switch up + go
to select TEXT -> go

Table 5.6. Possible APT descriptions of ROBART 1 (Note that the names attached to the schema sets are merely labels, and are not necessarily semantic attributes).

Inconsistency, then, has been identified by the mismatch between two sets — that of a "competent" user and that of the designer. The issue of the extra rule also still applies — from the point of view of user A, ROBART 1 has been designed inconsistently, as two rule schemata, are required to operate on one schema set. However, from the designer's point of view, there are two schema sets, and user A has made the error of trying to to treat all tasks as if they were similar.

There are, in fact, two separate issues here. The overall number of general rules is an absolute and set-independent way of predicting the learning burden — fewer implying less learning. However, that is a separate issue from what is addressed

here, which is concerned with how an interface's design is in accordance with, or contrary to, expectations derived from a user's prior mental contents. The example of "text" versus "shapes" that has been used so far, involves both a difference in grouping between designer and user, and an extra rule schema. In other cases, however, a user may simply group tasks differently from a designer, although both may expect the same overall number of general rules.

For example, consider a menu driven text editing program, containing the following commands: "PRINT" (prints out a document), "SAVE" (saves the user's work), "WORD COUNT" (counts the total number of words), and "SPELLING" (checks that words are spelled correctly). The designer has placed "PRINT" and "SAVE" on a menu labelled "FILE", and the commands "WORD COUNT" and "SPELLING" on a menu labelled "UTILITIES". The commands may be invoked by opening the appropriate menu (by clicking the mouse on the appropriate menu heading), moving the mouse over the name of the command to be invoked and clicking the mouse button.

Again consider two imaginary users, User A and User B. Both users accept that there are two sets of commands, however, they partition the tasks differently. User A thinks of "SAVE" and "PRINT" as file related tasks (say as "printing a file" and "saving a file"), and of "WORD COUNT" and "SPELLING" as utilities ("the word counting utility" and "the spelling checking utility"). However, User B thinks of "SAVE" and "WORD COUNT" as file related tasks ("saving a file" and "counting the number of words in a file"), and of "PRINT" and "SPELLING" as utilities ("the printing utility" and "the spelling checking utility"). This interface will be consistent for User A, but not User B. Although the designer and both users agree that there should be two general rules overall, the interface is inconsistent for User B, as he or she expects different settings from those of the designer (see table 5.7).

Task set
TASKSET = [invoke SAVE, invoke PRINT, invoke WORD COUNT, invoke SPELLING]

Schema sets (agent = designer)
FILE TASK = [SAVE, PRINT]
UTILITY TASK = [WORD COUNT, SPELLING]

Rule schemata (agent = designer)
to invoke FILE TASK -> click mouse on "FILE" + click mouse on FILE TASK
to invoke UTILITY TASK -> click mouse on "UTILITIES" + click mouse on UTILITY TASK

Schema sets (agent = User A)
FILE TASK = [SAVE, PRINT]
UTILITY TASK = [WORD COUNT, SPELLING]

Rule schemata (agent = User A)
to invoke FILE TASK -> click mouse on "FILE" + click mouse on FILE TASK
to invoke UTILITY TASK -> click mouse on "UTILITIES" + click mouse on UTILITY TASK

Schema sets (agent = User B)
FILE TASK = [SAVE, WORD COUNT]
UTILITY TASK = [PRINT, SPELLING]

Rule schemata (agent = User B)
to invoke FILE TASK -> click mouse on "FILE" + click mouse on FILE TASK
(for FILE TASK = SAVE)
to invoke FILE TASK -> click mouse on "UTILITIES" + click mouse on FILE TASK
(for FILE TASK = WORD COUNT)
to invoke UTILITY TASK -> click mouse on "UTILITIES" + click mouse on UTILITY TASK
(for UTILITY TASK = SPELLING)
to invoke UTILITY TASK -> click mouse on "FILE" + click mouse on UTILITY TASK
(for UTILITY TASK = PRINT)

Table 5.7. Possible APT notations for imaginary text editor.

Note, that for User B, the two set-independent general rules have become four rule schemata (two per set), whereas for user A they have become one schema per set.

The practical implication is that it is unlikely that inconsistency can be adequately described without a discovery procedure that captures both designer and user task groupings, and looks for mismatches. This implies that the identification of consistency (or inconsistency) is an empirical question.

5.1.6 Set compatibility

Set compatibility, then, can be defined as the extent to which a user's set partitionings are similar to those of the designer. From either agent's viewpoint, it is the similarity of the items that the other agent has grouped together, as compared to any competing affinity that they may have for other items not in the same schema set. Note that although neither previous work nor the studies reported in this thesis examine this directly, this view allows the possibility that items, even if grouped together, may be more or less similar, and might have a set of complicated "forces" pulling them more or less strongly towards each other. Thus, affinities between items may be of different strengths.

5.1.7 Rule compatibility

All of the above discussion relates to the identification of inconsistency by an extra rule. However, there is another issue, not tackled above, which is likely to have an effect on the usability of a system. It is the compatibility of a rule schema with the user's expectations. In the example in table 5.1, the rule schema for selecting a shape was to set the SHAPE switch up. For an American user of the machine this rule schema is likely to be more compatible with expectations than it would be for, say, a British user of the same machine. The reason is that in the USA in order to turn on an electrical appliance, such as a light, it is customary to have to set a switch to its "up" position, whereas in Britain it is customary to set the switch "down". Hence if the subject were to think in terms of "turning on" the option to select a circle, we might expect the schema to be closer to the expectations of an American user than to those of a British counterpart. So, if a system is not rule compatible difficulties will arise, not from an extra rule schema, but from the "unnaturalness" of the correct rule.

Returning to the text editor in table 5.7, from the point of view of User B, the task of invoking "WORD COUNT" is likely to be rule incompatible as well as set incompatible (User B would have expected the rule "to invoke WORD COUNT -> click mouse on "FILE" + click mouse on WORD COUNT"). However, he or she would probably regard invoking "SAVE" as rule compatible (but set incompatible), as he or she would be likely to expect a "file manipulation command" to appear on the "FILE" menu.

This issue was recognised by Payne and Green, with the inclusion of their "known-items" notation in TAG. A known-item appeared in the grammar where the investigator felt that the method for performing a task would be obvious to the user, based on what Payne and Green termed their "world knowledge". For example, if the command for moving a cursor up the screen were "UP", then this command name might be treated as a known-item (Payne and Green 1986). Howes and Payne's (1990) D-TAG — a formal representation of display based competence, included a similar notation "display-item". A "display-item" was included in the grammar where it was felt users' world knowledge would make the item to be selected from a display obvious to them.

Table 5.8 gives an APT like formalisation of the rules for selecting items with ROBART 1. This time, however, the agents are an American and a British user, and the formalisation has been extended to indicate user's a priori expectations about the task action-rules. So, whilst the interface is set incompatible for both British and American users, three of the four selection tasks — select line, select circle, and select square — would be rule compatible for the American.

Task set
TASKSET = [select line, select circle, select square, select text]

Schema sets (agent = designer)
SHAPE = [line, circle, square]
TEXT = [text]

Schema sets (agent = British user)
SHAPE = [line, circle, square, text]

Schema sets (agent = American user)
SHAPE = [line, circle, square, text]

Rule schemata (agent = designer)
to select SHAPE -> set SHAPE switch up + go
to select TEXT -> go

Expected rule schemata (agent = British user)
to select SHAPE -> set SHAPE switch down + go

Expected rule schemata (agent = American user)
to select SHAPE -> set SHAPE switch up + go

Table 5.8. Extended APT-style description of ROBART 1 for notional British and American users.

A system, then, will be rule compatible if the rule schema for a set of tasks is compatible with the user's expectations based on experience of the "outside world". Included in experience of the outside world are: experiences of doing similar tasks in daily life, experience of using similar machines, and knowledge of rule schema for doing other types of tasks on the same machine (i.e. tasks that the user regards as being in a different set). Note that a possible reason for inaccuracies found with production rule based predictions (e.g. Vossen et al. 1987) might be that the notation does not provide the mechanism for taking the suitability of rules into account. As with set compatibility, the identification of rule compatibility may require an empirical discovery procedure.

5.1.8 Internal and external consistency

When considering consistency, there is an apparently natural distinction that seems to occur, called by Kellogg (1987) internal vs. external consistency. The intuition is that a given item in an interface may independently have similarities to, or interference with, other items within the same interface (internal consistency), or with things experienced outside the interface (external consistency), for example other interfaces, or word uses in other areas of life. At first this seems to be the same distinction as between set and rule compatibility. On reflection, however, it can be seen that these distinctions do not correspond.

Whether an item (e.g. "text") is to be grouped with another item (e.g. "square") is a question of set compatibility, but can be affected by both internal and external issues: e.g. by whether the two are treated similarly with respect to other operations in the interface, but also by whether the user has experienced other programs in which they are treated similarly (in drawing programs) or differently (in word processors), and by whether that user has a strong disposition to regard text and squares as quite different semantic categories. Similarly rule compatibility can be affected by both internal and external considerations. The rule for one schema set might be similar to that of another schema set in the same program, or to that of a set of commands in another interface. Even considering only the special limited case of the consistency of a command with its menu title, rule compatibility will be affected both by what alternative menu titles are offered within the same interface and by external conventions for grouping in other programs (e.g. Macintosh conventions for the contents of the "FILE" menu).

Thus internal vs. external consistency, on examination, is a quite different issue from set vs. rule compatibility. Furthermore, it may not ultimately be a useful or even tenable distinction. How individuals group items within an interface depends on how they understand that interface, which itself depends on the understanding and conceptual categories which they bring to bear on that interface from outside. What, in the end, could "internal consistency" mean? If the particular interface were the only thing in a user's universe, then they would of course group things in the way the designer did, having no other ideas. There cannot be internal inconsistency unless users compare how the interface structures things with some other — that is external — system which gives rise to a feeling that the internal structure is somehow unnatural.

5.2 SUMMARY

Previous work on consistency has been reviewed, showing how ideas have evolved with successive treatments, with Reisner (1990) demonstrating that the issue is concerned with matches or mismatches between users' and designer's task groupings.

Two distinct types of consistency — set compatibility and rule compatibility — have been identified and defined. Meanwhile, the internal vs. external consistency distinction has been rejected.

Chapter 6

HOW MIGHT CONSISTENCY AFFECT USABILITY?

"It is almost a truism that consistency will make an interface easier to use", wrote Reisner (1981). In this chapter the issues of how set and rule compatibility might affect usability are addressed, and predictions are made about the effect of consistency on each of the components of usability outlined in chapter 2. These predictions formed a basis for the hypotheses tested in subsequent experiments.

After outlining the predictions, the assumptions on which they are based are discussed. Previous consistency related empirical work is then reported in support of these predictions.

6.1 PREDICTIONS

It was predicted that rule compatibility would have a large effect on performance during users' earliest interactions with an interface on a set of tasks, as at this stage they would still be dependent on world knowledge. However, it was thought that later on set compatibility would be more salient, as users would try to generalise across similar tasks, from their model of how the interface works. Below, more specific predictions are made about the effects of each type of consistency on each of the components of usability. The assumptions underlying these predictions are then discussed. Firstly, however, as a reminder, the various components of usability, with their definitions, are listed in table 6.1.

Guessability: The effectiveness, efficiency and satisfaction with which specified users can complete specified tasks with a particular interface for the first time.
Learnability: The effectiveness, efficiency and satisfaction with which specified users can achieve a competent level of performance on specified tasks with an interface, having already completed those tasks once previously.
EUP: The effectiveness, efficiency and satisfaction with which specified experienced users can achieve specified tasks with a particular interface.
System potential: The optimum level of effectiveness, efficiency and satisfaction with which it would be possible to complete specified tasks with an interface.
Re-usability: The effectiveness, efficiency, and satisfaction with which specified users can achieve specified tasks with a particular interface after a comparatively long time away from these tasks.

Table 6.1 Components of usability.

6.1.1 Set compatibility and usability

If the designer and user have partitioned tasks into sets in the same way, the user will only have one rule schema to learn and remember per task set — this should enhance learnability. EUP should also be enhanced by set compatibility, as the user shall not have more than one schema per set to choose between for a particular task — potentially resulting in reductions in time on task, and reducing residual errors.

The effect of set compatibility on a task's guessability is likely to be dependent on the level of familiarity that a user has with other tasks in the same set. For the first time user of a machine, or a user with no previous experience of other tasks in the same set, set compatibility should not affect guessability, as it is not connected with the issue of how easy a schema is to infer initially. Indeed, because set compatibilities only exist in the context of groups of tasks, it would not be possible for users to perceive them at this stage of interaction anyway. Thus, it does not seem possible that users could benefit from them. However, if a user has

previous experience of tasks which they regard as being in the same set, they may try to generalise a rule schema from these to the new task. The generalisation will only be valid if the user and designer have assigned the tasks to similar sets — hence, in this case, set compatibility would affect guessability.

The effect of set compatibility on re-usability is likely to be dependent on the length of time that a user has spent away from a task, and what he or she has been doing in that time. The less time users spend away from a task, the greater the influence of set compatibility may be. This is because they are more likely to try and remember schemata from their previous interactions, rather than resorting to inferring them from world knowledge. The less schemata there are to remember, the easier this is likely to be. Even if users haven't performed a particular task for a long while, set compatibility could still have an effect on re-usability if they had been performing other tasks from the same set in the meantime. This is because they may try to generalise a schema from these tasks to the one they are coming back to. However, if a user has been away from the interface as a whole, so that they have performed neither the task in question, nor any similar tasks for quite a while, the influence of set compatibility may diminish. Thus it might be expected that rule compatibility would be the more salient factor.

As set compatibility is not directly connected with the length or complexity of the action rules associated with a particular task, it should not affect system potential.

6.1.2 Rule compatibility and usability

Again, the effect of rule compatibility on guessability is likely to be dependent upon users' familiarity with other tasks in the same set. For users with no previous experience of such tasks, rule compatibility should make the rule easier to infer, thus enhancing guessability. However, those with a wider experience of having performed similar tasks are likely to try and generalise from these to the new task, rather than drawing on world knowledge. In such circumstances the effect of rule compatibility on guessability may be limited.

Rule compatibility should enhance learnability. If the rule for a particular task is compatible with expectations, the user will not have to "un-learn" an initially inferred incorrect schema before going on to learn the correct one. However, because rule compatibility does not affect the number of rules that the user must

remember and choose between for a set of tasks, it may not have a marked effect on EUP.

Time away from a task, and users' actions during this time are, again, likely to be factors in determining the effect of rule compatibility on re-usability. The longer users are away from a task, the more likely it seems that they will try and infer the schema from world knowledge, and thus the more important rule compatibility will be. However, if they have been performing similar tasks in the meantime, they may try and infer action rules from these, making rule compatibility of secondary importance.

As with set compatibility, there is no direct relation between rule length and complexity, and rule compatibility — thus it should not affect system potential.

6.1.3 Assumptions underlying predictions

As is the tradition in the consistency literature, predictions have largely been made on the basis of "common sense", rather then being tied to any particular model of, say, learning or information processing. Nevertheless, there are still some underlying assumptions. The first is that users will try to generalise a schema across tasks in what they regard as a set, rather than expecting each to have a different schema. Indeed, this is an assumption on which the whole of the consistency literature is based. If this were not the case, then set compatibility would not be expected to have any affect on usability at all, as inter-task transfer would not be a factor. It has also been assumed that users will try to apply world knowledge to new types of task. Again this assumption is not new. Payne and Green's (1986) "known-item" notation, discussed in the previous chapter, aims to quantify the benefits from this type of transfer. If this assumption were not valid, then rule compatibility would not be expected to have any effect.

An assumption which, perhaps, is more contentious, is that underlying the predictions about performance at EUP. It is predicted that here set compatibility will be important, whilst the effects of rule compatibility will be comparatively insignificant. This is based on the assumption that, for experienced users, it will be more easy to recall an unnatural rule, than to differentiate between several schemata applicable to (what the user regards as) a single schema set. The assumption was based on the idea that as the user's experience of a task set

increases, rules which initially seemed unnatural will become familiar. However, no amount of experience will change the number of schema required per set, so problems associated with this burden may remain longer.

The studies reported in subsequent chapters, testing the predictions are, in effect, also a test of the validity of these assumptions. Where there are deviations from the predicted results, the implications for the underlying assumptions are discussed.

6.2 EMPIRICAL SUPPORT FROM PREVIOUS WORK

In this section, some previous empirical work covering consistency related issues is reviewed in support of the predictions. Brief overviews of four studies are given, and the connections with set and rule compatibility are explained. Results are interpreted in the light of the multi-component usability framework. Three of the studies come from the command naming literature of the 1980s, whilst the other is a classical stimulus-response compatibility experiment.

6.2.1 Morin and Grant (1955)

This experiment was concerned with the spatial correspondence between the stimulus and response elements of a motor task. Earlier work by Fitts and Seeger (1953), had suggested that if direct correspondence (i.e. left most display with left most control etc.) was violated, a decrement in performance would result. Morin and Grant aimed to quantify this decrement.

The experimental apparatus consisted of two banks of lights and a bank of control switches, each arranged in a horizontal row. One bank of lights acted as stimuli, the idea being that as soon as one was illuminated, the subject had to turn on the equivalently placed light in the other bank by throwing one of the switches. Stimulus-response compatibility was manipulated according to whether or not the switch's position directly corresponded to the position of the stimulus light. By numbering the switches and lights from left to right, the experimenters were able to obtain a correlation coefficient (r) which they took as a measure of stimulus-response compatibility. So, for example, if each switch was directly opposite the light it controlled, then the correlation co-efficient would be 1. Conversely, had

the banks been arranged back to front, so that the switch controlling the far left light was on the far right of the row of switches etc., then the co-efficient would have been -1. All other combinations have associated correlation co-efficients somewhere in-between.

So, how does this relate to set and rule compatibility? Consider three possible arrangements represented in tables 6.2, 6.3, and 6.4.

Light number	1	2	3	4	5	6	7	8
Switch number	1	2	3	4	5	6	7	8

Table 6.2. Possible light / switch correspondence (r = 1).

Light number	1	2	3	4	5	6	7	8
Switch number	3	5	1	6	4	7	8	2

Table 6.3. Possible light / switch correspondence (r = 0.3)

Light number	1	2	3	4	5	6	7	8
Switch number	8	7	6	5	4	3	2	1

Table 6.4. Possible light /switch correspondence (r = -1)

For the first of these correspondences (table 6.2), the rule for turning on any particular light could be formalised thus:

to turn on light number X -> throw switch number X.

This arrangement makes the tasks of turning on the various lights set compatible as a general rule schema can cover them all. The rule also represents what would presumably be expected by most users — a direct spatial correspondence between lights and switches. Thus, this arrangement also gives rule compatibility.

The task action-rules associated with the last arrangement (table 6.4) also indicate set compatibility, a single rule schema covering them:

to turn on light number X -> throw switch number 9 - X.

However, here the arrangement does not give rule compatibility, as the rules are not as users would initially expect, but rather require the user to throw the switch in the "mirror image" position of the light they are concerned with.

The middle arrangement (table 6.3) gives neither set nor rule compatibility. There is no single rule schema that will cover the tasks, and none of the lights can be switched on using the most obvious switch.

So, the arrangements with co-efficients +1 and -1 appear to be set compatible, whilst the interface with co-efficient +1 is also rule compatible. No other arrangement would be either entirely set or rule compatible. Loosely, then, it appears that the size of the correlation coefficient may be related to an arrangement's set compatibility — as it is concerned with the position of the controls in relation to each other. Meanwhile, the sign of the coefficient appears related to rule compatibility, as it is concerned with the position of the controls relative to the lights — this might be regarded as a measure of how natural their position is.

The qualifier "loosely" has been used at the beginning of the previous statement, as it seems likely that these assumptions may only hold within certain limits. For example, the additional uniformity of an arrangement with an associated co-efficient of size 0.2, may not be sufficient to make it any more set compatible to a user than an arrangement with co-efficient size 0.1. Similarly, an arrangement with co-efficient +0.1 may scarcely be any more rule compatible than one with co-efficient -0.1.

Morin and Grant set their subjects a series of 9 blocks of twenty five trials each, and produced plots of time per block against block number for various light / switch arrangements. First task performance was included in time to complete the first block — it is difficult, therefore, to extract guessability values. The performance curves obtained started off at different levels and retained any comparative advantage or disadvantage throughout the experimental session. In terms of the multi-component usability framework then, these gave an idea of comparative learnabilities. If a criterion performance time were to be set, whatever it was, the curve which started lowest would be the first to pass through it. Thus,

this would represent the arrangement with the highest learnability. The curves did not level off during the experimental session, therefore the results did not reflect EUP. Examples of plots from the study are given in figure 6.1.

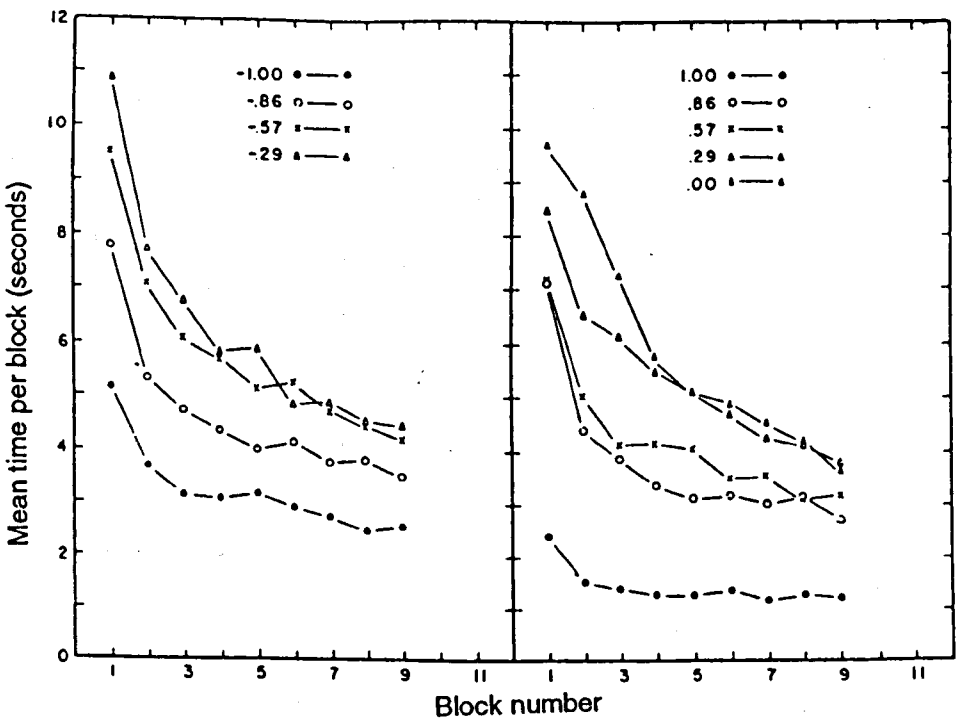


Figure 6.1. Examples of plots from Morin and Grant's study.

Morin and Grant found effects for both the size of the correlation co-efficient and its sign, with performance, as expected being clearly best on the arrangement with co-efficient +1 (set and rule compatible). This is in line with the predictions, which assert that learnability is likely to be affected by both set and rule compatibility.

6.2.2 Barnard, Hammond, Morton, and Long (1981)

This was an experimental investigation of argument ordering. The experiment included investigations of the effects of varying argument order within a set of commands, and of the effect of making argument order compatible with natural language.

The experimental scenario involved decoding a series of messages which appeared on a computer screen. This was done by typing in six commands, each of which performed one of the steps required. Each command consisted of a name followed

by two numeric arguments. One of these arguments was the number of the message to be decoded, the other would be dependent on the command being used. Four command languages were used, examples of which are given in table 6.5. Compatibility with natural language was manipulated according to whether or not the position of the direct object of the command verb was the first argument — having it first was thought to be more natural language compatible.

Language 1 (direct object first)	SEARCH, file number, message identifier	REPLACE, message identifier, code number
Language 2 (direct object second)	SEARCH, message identifier, file number	REPLACE, code number, message identifier
Language 3 (message identifier first)	SEARCH, message identifier, file number	REPLACE, message identifier, code number
Language 4 (message identifier second)	SEARCH, file number, message identifier	SEARCH, code number, message identifier

Table 6.5. Examples of the experimental command languages.

In terms of consistency, these different languages appear to differ with respect to both set and rule compatibility. Set compatibility in this context relates to whether or not the position of the message identifier is constant between tasks. Thus, tasks performed with languages 3 and 4 were set compatible, whilst those performed with languages 1 and 2 were not. Rule compatibility, meanwhile, is to do with whether argument order is compatible with natural language — although the authors don't comment on subjects' previous computing experience, it was assumed that knowledge of natural language would be a significant part of the world knowledge which subjects brought to bear on the tasks. All tasks performed with language 1 would thus be rule compatible, as the direct object was always entered first. With language 2 all tasks were rule incompatible, as the direct object was always second. Some of the tasks performed with each of languages 2 and 3 were rule compatible, as the direct object is sometimes first in these languages, but not always.

In an earlier study, reported in the same paper, Barnard et al. investigated their premise that having the direct object first was more natural. A group of subjects was presented with a questionnaire containing a series of twelve pairs of

commands. Each member of a pair was for performing the same task, the only difference between them being that for one the direct object was the first argument, whilst for the other it was the second. The commands were similar to those used in the main study reported here. Subjects were asked to rate which of each pair they regarded as more appropriate. For ten out of the twelve pairs, most subjects rated the commands with direct object first as being most suitable. A binomial test showed these preferences to be statistically significant for all ten of the pairs ($p < 0.1$, $n = 68$, 1 tailed test). For the other two commands most subjects chose the command with the direct object as the second argument, however, effects were not statistically significant for either of these. On average, commands with the direct object first were rated as more appropriate by 67 % of subjects. Hence, it seems reasonable to expect that having to enter the direct object first would be closer to the a priori expectations of the subjects in the main study. Table 6.6 summarises the set and rule compatibility of tasks performed with each language.

LANGUAGE	SET COMPATIBLE?	RULE COMPATIBLE?
Language 1	No	Yes
Language 2	No	No
Language 3	Yes	Partially
Language 4	Yes	Partially

Table 6.6. Set and rule compatibility of tasks performed using the experimental languages.

Each subject decoded a series of ten messages, using one of the four languages. Decoding of the first two messages acted as a training phase, instructions being displayed to subjects, but from then onwards subjects attempted the tasks on their own — there were, then, eight experimental trials.

The statistical analysis of the results performed by the authors was based on combined performance over all eight trials. This showed effects for both the position of the direct object, and for varying argument order. However, because performance over the trials was combined, it was not clear which components of usability these figures represented.

Of more interest, in this context, were plots showing performance trial by trial using the different languages. Several plots were presented showing performance in terms of various experimental measures. Two of these — time (between being

requested to perform a task and typing the first letter of the command name), and instruction requests — appeared to be levelling off by the end of the session (see figures 6.2 and 6.3). It was possible, then, to make comparisons of learnability and EUP from these.

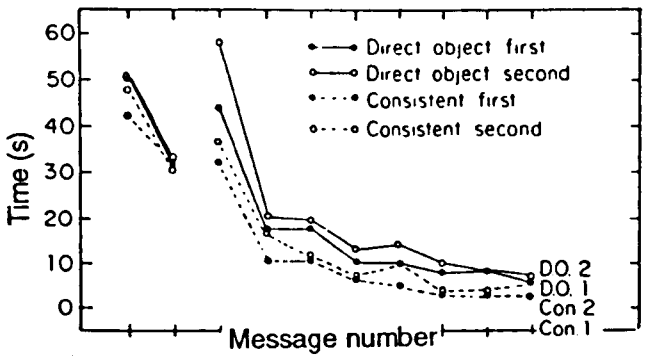


Figure 6.2. Performance by trial in terms of time.

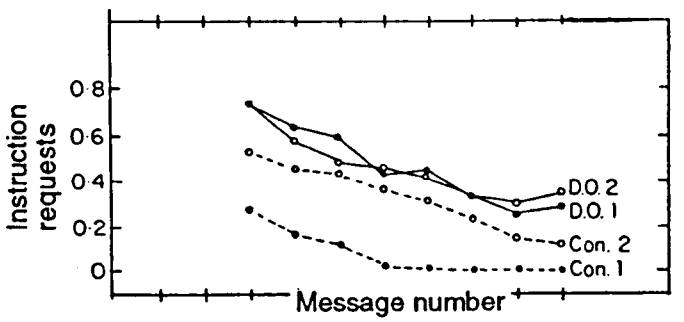


Figure 6.3. Performance by trial in terms of instruction requests.

On the last few trials there was little difference in performance between tasks performed using language 1 and those performed using language 2. This suggests little effect for rule compatibility at EUP, as predicted. However, performance with languages 3 and 4 was markedly better, showing, as predicted, an effect for set compatibility at EUP. Earlier in these plots, and in curves showing performance in terms of other measures, which did not appear to have levelled off, performance was best with languages 3 and 4, indicating an effect for set compatibility on learnability. On one of these curves (errors), as well as with the time curve (figure 6.2) performance appeared to be better using language 1 than with language 2, indicating the predicted effect for rule compatibility on learnability, whilst on the other plots there were no clear differences between these two.

6.2.3 Carroll (1982)

Carroll investigated the effect of a property of command names called congruence. This property, formalised by Carroll (1980), is concerned with whether or not command names suitably reflect the relationships between the tasks which they are needed for.

The context of the experiment was learning a command language to control the actions of an imaginary robot. Four different types of language were used in the experiment: a congruent hierarchical language, a congruent non-hierarchical language, a non-congruent hierarchical language, and a non-congruent non-hierarchical language. (Hierarchicalness was another property being investigated in this study, however the property has no obvious relationship with consistency, and thus is of little interest here).

As an example of congruence, consider the following non-hierarchical commands. The command for making the robot turn left was "LEFT". For turning the robot to the right, the command in the congruent language was "RIGHT", whilst in the non-congruent language it was "TURN". The difference, then, is that, in the congruent language, opposite tasks require opposite command names — "LEFT" and "RIGHT", whereas in the non-congruent language command names have a less obvious relationship.

In the hierarchical paradigms, congruence was manipulated via the operator name in the command string. Again, consider turning the robot to the left and right. The paradigm for left turning was "MOVE ROBOT LEFT". In the congruent condition the paradigm for right turning was "MOVE ROBOT RIGHT", whilst in the non-congruent condition it was "CHANGE ROBOT RIGHT".

In terms of consistency, congruence is a set compatibility issue — are similar tasks performed using similar commands? Consider the hierarchical command paradigms for left and right turning. A congruent pair could be represented using the following formalism:

to turn robot in direction X -> type MOVE + type ROBOT + type X,

where the set X = [LEFT, RIGHT].

With the non-congruent paradigms however, two schemata would be needed for set X:

to turn robot in direction X -> type MOVE + type ROBOT + type X (for X = LEFT)

and

type CHANGE + type ROBOT + type X (for X = RIGHT).

The non-congruent paradigms, then, are not set compatible.

With the non-hierarchical paradigms, capturing the set incompatibilities via a formalism is less straightforward. What is needed is a notational device similar to one devised by Payne and Green (1986) for their task-action grammar. The notation, which aims to capture featural similarities, is best illustrated by means of an example. Consider the rules for turning the robot left and right in the non-hierarchical conditions. In the congruent condition, the action-rules could be represented thus:

to turn robot in direction X -> type F (LEFT),

for X = [left, right].

The F() notation means the "appropriate equivalent of" the feature in brackets. So, if the robot is to be moved to the right, the appropriate equivalent of "LEFT" would be "RIGHT". However, for the non-congruent interface, such a formalisation would not work as a means of reducing the rules for performing the tasks to a single schema, as "TURN", could not reasonably be described as an equivalent of "LEFT".

The subjects worked their way through booklets. A booklet contained a list of commands for controlling the robot in one of the four experimental languages. Different subjects' booklets contained different languages. After reading through the commands (as part of a rating task) subjects performed an open recall test — writing down as many of the command names and functions as they could remember. Next, they were asked to right down the command sequence required

to get the robot to perform a particular task. A prompted recall test was then given — subjects being asked to put definitions to a scrambled list of command names, before a second "task" was set. There was then a second open recall test.

Again, all experimental measurements were taken after the subjects had had prior exposure to commands — therefore guessability was not being measured. Results, then, would probably be indicative of learnability, and possibly EUP (depending on whether performance had steadied by the end of the experimental session). Either way, as set compatibility was predicted to affect both learnability and EUP, an effect for congruence would be expected.

Indeed, this was what the results showed — a statistically significant advantage for the congruent languages, which was maintained throughout the experimental session. (The only effect found for hierarchicalness was on the final recall test where, contrary to Carroll's expectation, results showed an advantage for non-hierarchical paradigms).

6.2.4 Payne and Green (1984)

This was an investigation of the organisation of command languages. Subjects with little computing experience were presented with a series of command / name pairings to learn, and were then asked to recall them.

Four experimental word processing languages were used, each containing commands for performing the same 26 functions. Each language relied on the use of keys labelled "ctrl" and "meta" along with another key. Samples of each are listed in table 6.7.

TASK	LANGUAGE 1	LANGUAGE 2	LANGUAGE 3	LANGUAGE 4
move pointer to end of file	ctrl - F	ctrl - M	ctrl - >	meta - >
move pointer to beginning of file	meta - F	meta - M	ctrl - <	meta - <
move pointer to end of line	ctrl - L	ctrl - N	ctrl - Z	ctrl - E
move pointer to beginning of line	meta - L	meta - N	ctrl - A	ctrl - A
view next screen	ctrl - V	ctrl - C	ctrl - V	ctrl - V
view previous screen	meta - V	meta - C	ctrl - ^	meta - V

Table 6.7. Samples of Payne and Green's (1984) experimental languages.

Language 1 uses a uniform organising principle throughout, with ctrl associated with moving forwards, and meta associated with moving backwards. It also has sensible mnemonic symbols, for example, F for file, and L for line. Language 2 uses the same organising principles, but randomly chosen non-mnemonic symbols. Language 3 has no overall organising principles, but rather all the commands are pre-fixed with ctrl. However, within the constraints of using single symbols, the best possible mnemonic codes have been used (in the judgement of the authors). Language 4 was a subset of a real word processing language called "Mince". Although the mnemonic codes are sensible, the language does not appear to be structured in accordance with any clear organising principle.

In this context, set compatibility is concerned with the organising principles used. Intuitively, there appear to be two ways in which subjects might group tasks — by direction of movement or by degree of movement. Thus tasks can either be set compatible or set incompatible in these two ways. Language 1 is set compatible in both ways, ctrl is always used where forward movement is required and meta where there is backwards movement. Similarly, F is always used if moving through a file, L if through a line, and V if through a page. Language 2 is also set compatible in both of these ways. Language 3 is set compatible in the first way —

ctrl always being used for forward and backward movement, but not in the second way. Language 4 is not set compatible in either way, whilst meta is used in moving to the beginning of a file, ctrl is used to move to the beginning of a line. Equally, the additional keystrokes show no clear organising principle.

Rule compatibility, here, is concerned with whether or not all or part of the command string could be predicted from world knowledge. Language 1 and language 4 should be comparatively rule compatible, then, as they utilise sensible mnemonic symbols. Language 3 also does to some extent, but only within the constraints imposed by being able to use each symbol only once. This should make it less rule compatible than languages 3 and 4, but more so than language 2, where non-mnemonic symbols are used totally at random.

Set and rule compatibilities of tasks performed with each of the languages are summarised in table 6.8.

LANGUAGE	SET COMPATIBLE?	RULE COMPATIBLE?
Language 1	Yes	Yes
Language 2	Yes	No
Language 3	Partially	Partially
Language 4	No	Yes

Table 6.8. Set and rule compatibility of tasks performed using the experimental languages.

Each subject was presented with 26 cards each containing a command / name pair. Subjects were allowed to lay the cards out as they pleased, and were given twelve minutes to study them. They were then set a free recall test, being asked to write down as many command / name pairs as they could remember. No time limit was put on this test, subjects being allowed to continue until they ran out of ideas. A prompted recall test was then administered. Here, subjects were issued with sheets listing the commands, and had to write the appropriate name next to each. Again there was no strict time limit. Subjects then scored their own tests — they were specifically asked to try and learn from this. Another free recall test was then administered, followed by a second prompted recall test.

Because subjects had seen the commands prior to any experimental measures being taken, guessability was not addressed. The results, then, should be

indicative of learnability and possibly, if performance had steadied by the end of the experimental session, EUP.

On the first prompted recall test a statistically significant advantage was found for language 1 over all of the others. This is not surprising in the light of the predictions, as it is the only language exhibiting both set and rule compatibility — both factors predicted to affect learnability. Performance with language 4 was significantly worse than with any other language in the first tests. However, this result was difficult to interpret in the light of the predictions as the language exhibited one type of consistency, but not the other.

By the second set of tests, the advantage of language 1 over languages 2 and 3 has been lost. In terms of the predictions, this suggests that EUP may have been reached, so that rule compatibility no longer affects performance. This would imply that performance should still be worst with language 4, as it is the only one which does not exhibit set compatibility. Indeed, statistically significant performance differences were found with this language compared with the others, on both the second free recall and prompted recall tests.

6.2.5 Comment on the studies and possible further reading

The studies reported above have addressed issues connected with both set and rule compatibility. Three of the four studies came from the command name literature, a comparatively rich source of experiments addressing consistency related issues (see also, for example, Black and Moran 1982, Scapin 1982, Payne and Green 1989). The particular studies reported were chosen as (apart from Carroll 1982) they addressed both set and rule compatibility issues, and (apparently) measured both learnability and EUP, rather than taking "one-point" usability measures. Unfortunately, however, no studies where measurements reflecting guessability or re-usability could be found.

Morin and Grant's stimulus-response experiment was chosen as it demonstrates that these effects can apply outside the context of learning and remembering command names. It also indicates a possible connection with stimulus-response compatibility issues, and the vast accompanying literature.

There are other related studies which may be of interest, which have not been reported here, either because they only covered issues related to one type of consistency, or because performance was only measured on a one off basis. These include: Reisner 1981 (graphics system, used as example in previous chapter), Rosenburg 1982 (suggestiveness of command names, a rule compatibility issue), Norman and Fisher 1982 (looked at alphabetic keyboards — these should be set compatible as the relation of keys to each other is determined by coherent framework, and rule compatible as this framework forms part of users' world knowledge), and Kellogg 1987 (manipulated the method of command activation in an automated office system, addressing set compatibility issues).

6.3 SUMMARY

Predictions about the effects of set and rule compatibility on the various components of usability have been outlined, and the assumptions underlying these discussed. Broadly, these reflect the idea that rule compatibility will be more important during earlier interactions, with set compatibility taking effect as users' experience increases. Previous empirical work related to set and rule compatibility issues has been reported, providing support for predictions about the effect of consistency on learnability and EUP.

Chapter 7

APPLYING THE THEORY TO MENU COMMANDS

In the previous chapter some empirical support for the predictions about the effects of set and rule compatibility on usability was provided in the context of previous empirical work. However, this was based on post-hoc interpretations of studies originally conducted for other purposes. This meant that, in the context of the issues addressed in this thesis, methodologies were not ideal, and there was often ambiguity about which component of usability was reflected by the results. Further, even accepting that these studies have been re-interpreted correctly, learnability and EUP were the only components covered.

This chapter reports the first of five studies, specifically aimed at investigating the effects of set and rule compatibility on the various components of usability. The study investigated the usability of an interface with respect to four different types of task (types A, B, C, and D), chosen so that they would cover all four possible combinations of set and rule compatibility (see table 7.1)

	RULE COMPATIBLE	RULE INCOMPATIBLE
SET COMPATIBLE	Type A	Type B
SET INCOMPATIBLE	Type C	Type D

Table 7.1. Set and rule compatibility of experimental tasks.

The predictions made about the effects of consistency on usability, as reported in the previous chapter, led to the following experimental hypotheses (table 7.2). Here A > C, for example would mean that it was expected that type A tasks would be more usable than type C tasks with respect to the component in question (i.e. can be performed at less cost to the user). Meanwhile A = C would mean that the two are expected to be equally usable.

COMPONENT OF USABILITY	EFFECT OF SET COMPATIBILITY	EFFECT OF RULE COMPATIBILITY
Guessability	A = C, B = D	A > B, C > D
Learnability	A > C, B > D	A > B, C > D
EUP	A > C, B > D	A = B, C = D
System potential	A = C, B = D	A = B, C = D
Re-usability	A > C, B > D	A > B, C > D

Table 7.2. Experimental hypotheses.

Note that the predictions about guessability are based on the assumption that subjects would initially be naive to the experimental interface, and would thus attempt to infer schemata from their experience outside this program.

The type of task used to test these predictions was the invocation of word processing commands from pull down menus (where both menus and commands had textual names). In this context, the rule for invoking a command has to specify which menu to open, and rule compatibility is concerned with whether the name of the command seems to the user to suggest the name of the necessary menu. Set compatibility here concerns whether commands on a single given menu seem to the user to go together, irrespective of whether any or all of them seem to go with the menu title. As an example, consider a menu headed "WINDOW", containing commands named "SMALLER FONT SIZE", and "LARGER FONT SIZE" — this is taken from one of the experimental interfaces (described in more detail in the "Apparatus" section of the method). Invoking these commands would be set compatible, as two similar commands are together, and hence the action-rules for their invocation could be represented by a single rule schema:

to invoke X -> Open WINDOW menu + select X,

where X = [SMALLER FONT SIZE, LARGER FONT SIZE].

However, this would not be rule compatible as a user would not expect to find these commands on a menu called "WINDOW".

In this context, then (which corresponds to a very common user interface design issue), consistency maps on to a situation similar to experiments on word pair associations.

Much of this work compared meaningless to meaningful names, showing an advantage for the latter, as might be expected. More recently, some work has addressed the choice of names for HCI tasks as opposed to word-word associations (Landauer, Gallotti, and Hartwell 1983). The study reported here is distinct from Landauer et al.'s work, because it compares word-word associations, where associations between command names are compared and contrasted to associations between command names and menu titles. In doing so it relates these to a theory of consistency, and there is no reason to think that this type of task is not fully representative of general consistency issues. For instance pictures might be used instead of text (indeed this was explored in a study reported in a later chapter). In fact menus share not just a title but a spatial position, which subjects may also begin to learn.

The rules in this case all have the same general pattern of: "open a specific menu, move down to select a command", while in other interfaces they can differ not only in the menu title but in the number, sequence, or type of the component actions (again this is explored in a later study). These variations all have equal status in the theory, but there is no reason to expect them to affect performance differently. The simplification of having rule schemata differ only in the menu title was chosen for this study so that time measurements of actual subject behaviour would not be confounded by necessary differences in duration of the physical movements, but instead would reflect only differences in mental processing.

Presumably the *amount* of set or rule compatibility varies. For instance "SMALLER FONT SIZE" and "LARGER FONT SIZE" might seem more similar, and be associated with each other faster or more persistently, than, say, "NEW WINDOW" with "ZOOM WINDOW". However, this issue is not addressed here (or in the consistency literature as a whole), and the predictions are couched in terms of inequalities and approximate equalities, not quantities or proportions. Similarly, in the context of this experiment it should be sufficient for two names to be compatible in the sense of being distinctly more similar to each other (in both words and meaning) than to any alternative.

A further related issue is that of what, exactly, users would normally remember about a command. It might be expected that, in general, users remember a command's meaning or effect or role in a task, its name or label as displayed for recognition by the user, its selector or keyboard accelerator (e.g. command-B for

Bold, or "3" for item 3 on menus where numerical selectors are used) where commands are selected by some indirect means, and perhaps its position, especially where menus are selected by pointing. All of these in principle will contribute to consistency effects. Provided however that other components do not have striking similarity aspects, then an experiment relying (as this one does) on a single component (the name in this case) should work.

It was decided to use numerical selectors for opening the menus, rather than mouse pointing, partly to avoid having to train the subjects with the mouse (all had had prior experience with keyboards, few with mice), but mainly to increase the time penalty of making a wrong menu choice and so motivate the subjects to learn where the commands were. (There is some evidence that users do not bother to learn aspects of an interface if the information is available via perception when needed. Even being able to "cruise" through the menus looking for a target, reduces the need to learn locations). In this experiment, then, consistency effects were confined to the textual names. Any learning of position or menu selector numbers would not be likely to make some associations more compatible than others, although in principle it cannot be ruled out that there is the possibility of some user having a strong prior experience of, say, the "Zoom window" command always being in the left-most menu, or always being in a menu whose selector is "3".

7.1 THE STUDY

7.1.1 Apparatus

A menu driven word processing package, Microsoft Word 4 for the Apple Macintosh, was adapted for the experiment. Initially, all but eight commands were removed from the menus. Those that remained were pairs which, it was assumed, users would group as being members of the same set (table 7.3), since they had both words and their meaning in common.

"Natural" menu heading	Experimental command	Filler command
FORMAT	COPY FORMATS	FIND FORMATS
FONT	SMALLER FONT SIZE	LARGER FONT SIZE
DOCUMENT	MOVE TO START OF DOCUMENT	MOVE TO END OF DOCUMENT
WINDOW	NEW WINDOW	ZOOM WINDOW

Table 7.3. Commands used in experiment, and their associated menu headings.

For each pair of commands, the activation of one of the pair was to be an experimental task, and the activation of the other a filler task. The commands were placed on menus in such a way as to ensure that activation of the experimental commands would produce tasks of types A, B, C and D. Figure 7.1 illustrates one of four menu configurations used in the experiment.

FORMAT	FONT	DOCUMENT	WINDOW
COPY FORMATS	MOVE TO END OF DOCUMENT	MOVE TO START OF DOCUMENT	LARGER FONT SIZE
FIND FORMATS	NEW WINDOW	ZOOM WINDOW	SMALLER FONT SIZE

Figure 7.1. An experimental menu configuration

With this configuration, the task of activating "COPY FORMATS" would be both set and rule compatible (type A). The rule for activation ("Open FORMAT menu + select COPY FORMATS") is similar to that for "FIND FORMATS" (i.e. utilises the same rule schema and hence is set compatible), and is (it was assumed) the rule the user would most likely expect (hence rule compatible). Activating "NEW WINDOW" is neither set nor rule compatible (type D), as the activation rule ("Open FONT menu + select NEW WINDOW") is not similar to the rule for activating "ZOOM WINDOW" ("Open DOCUMENT menu and select Zoom window"), and hence is set incompatible, nor is it what the user might have expected (which would be"Open WINDOW menu and select NEW WINDOW"), hence it is rule incompatible. Activating "MOVE TO START OF DOCUMENT" is set incompatible, but rule compatible (type C), whilst activating "SMALLER FONT SIZE" is set compatible, but rule incompatible (type B). Formalisations of the action rules associated with each invocation are listed in table 7.4. The

experimental tasks are shown in italics. Note that there are in fact seven menus included in Microsoft Word. The other three were left empty here — thus their headings acted as distractors in this experimental context.

Task set
TASKSET = [*invoke "COPY FORMATS"*, invoke "FIND FORMATS", *invoke "SMALLER FONT SIZE"*, invoke "LARGER FONT SIZE", *invoke "MOVE TO START OF DOCUMENT"*, invoke "MOVE TO END OF DOCUMENT", *invoke "NEW WINDOW"*, invoke "ZOOM WINDOW"]

Schema sets (agent = designer)
SET W = [*COPY FORMATS*, FIND FORMATS]
SET X = [*NEW WINDOW*, MOVE TO END OF DOCUMENT]
SET Y = [*MOVE TO START OF DOCUMENT*, ZOOM WINDOW]
SET Z = [*SMALLER FONT SIZE*, LARGER FONT SIZE]

Rule schemata (agent = designer)
to invoke W -> Open FORMAT menu + select W
to invoke X -> Open FONT menu + select X
to invoke Y -> Open DOCUMENT menu + select Y
to invoke Z -> Open WINDOW menu + select Z

Schema sets (anticipated) (agent = user)
SET P = [*FIND FORMATS*, COPY FORMATS]
SET Q = [*SMALLER FONT SIZE*, LARGER FONT SIZE]
SET R = [*MOVE TO START OF DOCUMENT*, MOVE TO END OF DOCUMENT]
SET S = [*NEW WINDOW*, ZOOM WINDOW]

Rule schemata (agent = user)
to invoke P -> Open FORMAT menu + select P
to invoke Q -> Open FONT menu + select Q
to invoke R -> Open DOCUMENT menu + select R
to invoke S -> Open WINDOW menu + select S

Set compatibility
P matches W
Q matches Z
R has no match
S has no match

Rule compatibility
Rules for invoking "*COPY FORMATS*", "FIND FORMATS", and "*MOVE TO START OF DOCUMENT* " match.

Table 7.4. Formalisation of task action-rules associated with interface illustrated in figure 7.1.

A summary of tasks by type, for this configuration is give in table 7.5.

TASK TYPE	EXPERIMENTAL TASK	PLACED UNDER HEADING	FILLER TASK UNDER HEADING
A	invoke COPY FORMATS	FORMAT	FORMAT
B	invoke SMALLER FONT SIZE	WINDOW	WINDOW
C	invoke MOVE TO START OF DOCUMENT	DOCUMENT	FONT
D	invoke NEW WINDOW	FONT	DOCUMENT

Table 7.5. Tasks by type (see table 7.2) for the configuration illustrated in figure 7.1.

In all, four, separate menu configurations were used. This was in order to balance the assignment of experimental tasks to task types across interfaces. So, for example, whilst, in the configuration illustrated "invoke COPY FORMATS" was a type A task in another this would be a type B task, with, say, "invoke NEW WINDOW" being of type A (see appendix 7.1).

7.1.2 Subjects

Thirty-six students from the faculties of Arts, Social science, and Science at the University of Glasgow participated in the study. Each subject was classified as belonging to one of four groups, depending on gender and previous experience with word processors (see table 7.6).

	Male	Female
Previous experience	n = 4	n = 8
No previous experience	n = 8	n = 16

Table 7.6. Experimental subjects.

In this study "experienced" subjects were those who had experience both of word processing (task domain knowledge), and of the Macintosh operating environment (operating environment knowledge) — because the word processing package used

in the experiment had been extensively modified, no subject would have knowledge of the application program. "Inexperienced" subjects had used neither a word processor, nor a Macintosh computer before.

7.1.3 Experimental design

There were two experimental sessions. During the first, each subject performed a series of twelve trials. A trial consisted of a block of the four experimental tasks, followed by the four fillers. The order in which the experimental tasks were set was balanced for task type, both between subjects (so, if one subject did, say, task type A first in his or her first trial, then another might do, say, type B first), and within subjects (so that if, for example, a subject were to do task type C first during the third trial, he or she may do, say, type D first during he fourth trial). This was done via a balanced Latin-square design (Elmes, Kantowitz, and Roediger 1981). This is described in more detail in appendix 7.1. The order in which the filler tasks were set was determined at random by "pulling numbers out of a hat".

Within each subject group, equal numbers were allocated to use each menu configuration. Allocation, within this criterion, was done randomly by "pulling names out of a hat". About twenty four hours after the first experimental session, subjects returned to repeat the experimental tasks once each. Performance on this trial was taken to be indicative of re-usability.

Performance was measured by number of errors, and time on task. An error was said to have occurred when a subject opened an incorrect menu, in the process of command activation (as this would mean that he or she was attempting to use an incorrect rule to activate a command).

7.1.4 Procedure

After reading an overview of the experiment (appendix 7.2), and completing a personal details interview, subjects were shown how to activate a command. This was done by using a combination of the "apple" key[1] the "tab" key and a number to open a menu, using cursor keys to move up and down the menu, and pressing

[1]This is a key specially designed for command invocation with the Macintosh.

return when the appropriate command was highlighted. The subjects practised command activation using a made up menu, which they would not be using in the experimental session. Thus they had no practice with the experimental menu configurations before the experimental trials started. This was important, as performance on the first trial was to be used as an indicator of guessability. Each subject practiced, until they had made a minimum of ten command activations. If, by this time, he or she was still not "up to speed" (either in the subject's opinion, or the opinion of the experimenter) with the physical mechanics of activation, practice continued until both subject and experimenter were satisfied that he or she was.

The first task was then set — the experimenter reading out the name of the command to be activated. As soon as a subject completed a task they were set another straight away. A separate text file was displayed on the screen for each trial. Subjects were told that invocation of the experimental and filler commands caused manipulations of these files. However, the display gave little feedback as to the effects of each command, and it was assumed that subjects would make associations between commands and menu titles on the basis of their textual names. However, even if the functions of commands were to be considered, it was assumed that these associations would have been similar. Between trials there was a short gap whilst the experimenter removed one file from the display (closed it) and opened another. The experimenter followed a written protocol throughout the experiment (appendix 7.3), to ensure that subjects were treated fairly uniformly. The first session lasted about an hour.

In the second session subjects were set one more trial, including only the experimental commands.

7.2 RESULTS

The overall profiles of number of errors and time on task, by trial are illustrated in figures 7.2 and 7.3 respectively.

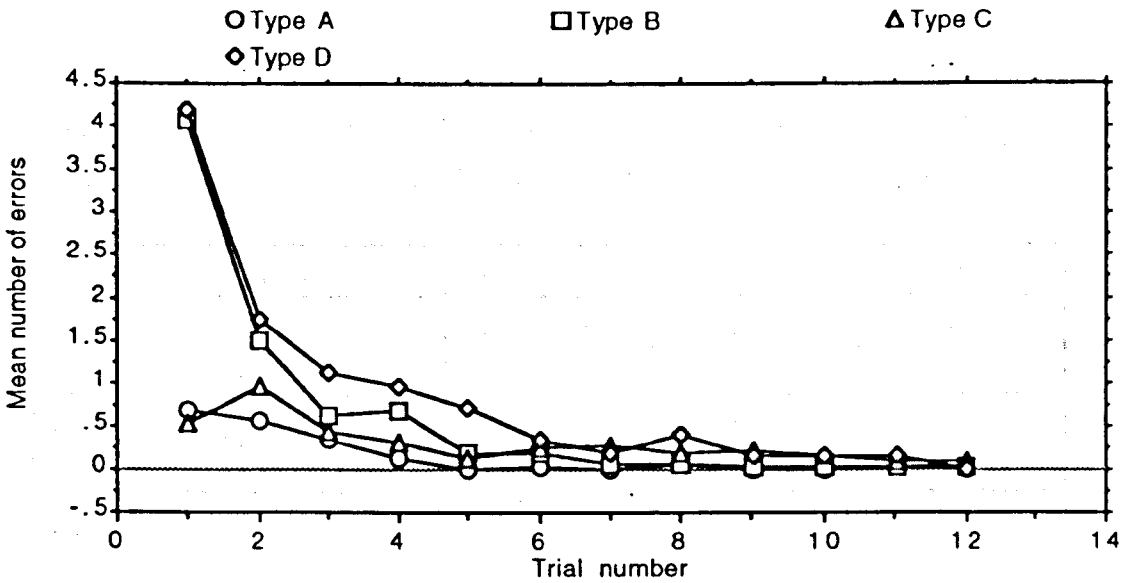


Figure 7.2. Mean number of errors per trial for each task type.

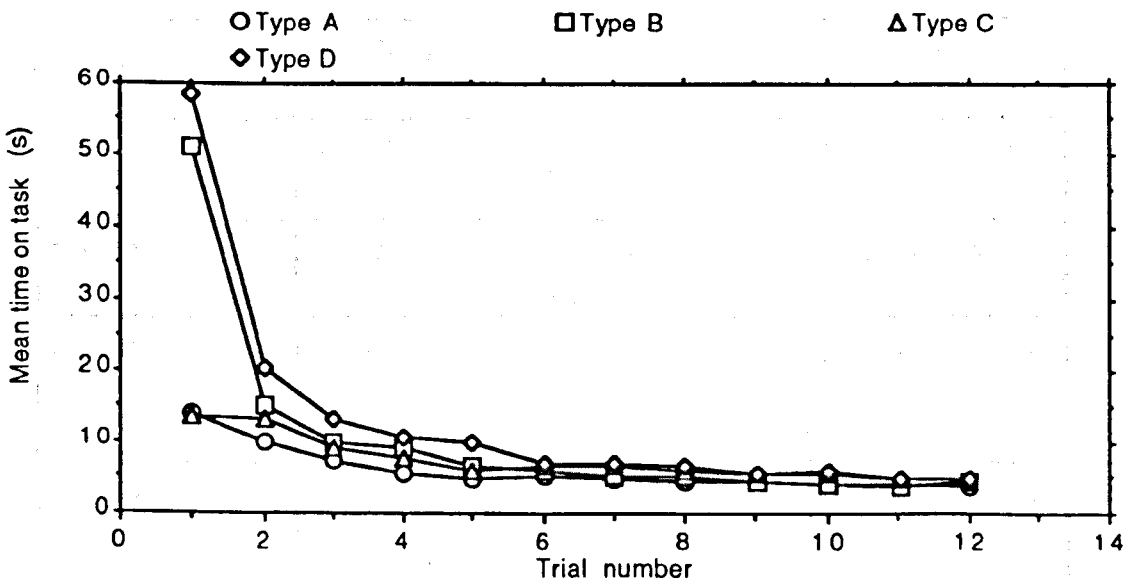


Figure 7.3 Mean time on task per trial for each task type.

7.2.1 Guessability

The number of errors made, and time taken to complete tasks, during the first trial were taken as being representative of guessability. The less the errors and time the higher the guessability. Mean values for each type of task are given in table 7.7.

Measure	Type A	Type B	Type C	Type D
Errors	0.67 (sd = 1.30)	4.06 (sd = 2.69)	0.53 (sd = 1.18)	4.19 (sd = 3.07)
Time on task	14.0 s (sd = 11.00 s)	51.2 s (sd = 43.6 s)	13.6 s (sd = 10.7 s)	58.4 s (sd = 53.9 s)

Table 7.7. Mean guessability values.

The experimental predictions were checked, by carrying out Wilcoxon signed rank tests (WSR) to check for statistically significant differences between results. Table 7.8 illustrates that the results were as predicted. Remember that predictions such as A > C means it was predicted that users would be able to complete type A tasks at less cost than type C (i.e. taking less time and making less errors).

Comparison	Prediction	Errors	Time on task (s)
A vs C	A = C	NS, T = 40 (Tc = 17, n = 13)	NS, T = 301
B vs D	B = D	NS, T = 302 (Tc = 195, n = 35)	NS, T = 315
A vs B	A > B	$p < 0.01$, T = 28 (Tc = 182, n = 34)	$p < 0.01$, T = 59
C vs D	C > D	$p < 0.01$, T = 15.5 (Tc = 170, n = 33)	$p < 0.01$, T = 36

Table 7.8. Comparison of guessability between task types. Where tied values have been excluded, so that $n < 36$, this is indicated.

7.2.2 Learnability

The number of trials, subsequent to the first, that subjects performed before making their final error on a particular type of task was taken as representative of learnability for that task — the fewer trials, the higher the learnability. (The first trial was not considered here, as it had already been considered in the analysis of guessability). In the evaluation of the VCR, reported in chapter 4, subjects only had to go two consecutive trials without error. However, the task set there was far more complex, so that requiring error free performance until the end of the session might have led to distortions due to one off slips. However, this should not be a factor with the comparatively simple tasks set here, where any errors made would relate directly to what is being investigated — do subjects look for the commands on the appropriate menu? In terms of time, a value of learnability was taken from the number of the first of two consecutive trials, subsequent to the first trial, in which subjects completed a task in under six seconds. A six seconds completion time was chosen to represent a "competent" level of performance, as most subjects managed this on all types of task during the experimental session, and thus it facilitated inter-task comparisons. So, for example, if a subject completed task type C in 5.5 s on trial 7 and 5.3 s on trial 8, then he or she would be ascribed a learnability value of 6 trials (remember, the first trial was excluded from the analysis). The reason for using two consecutive trials as a criterion was to avoid a one-off "lucky strike" distorting the results.

Where a subject failed to satisfy the learnability criteria in terms of a particular measure, for a particular type of task, a default value of twelve trials was set. This value was chosen on the basis of assuming that, were subjects to have continued, they would satisfy the criterion in question on their next trial — thus these default values provide lower bound learnability values for those tasks. Clearly, setting lower bounds means that estimates of inter-task differences may be under-played. However, these values can be used in non-parametric statistical analyses, such as the WSR test, the outcome of which will thus give a conservative estimate of whether differences are statistically significant. The most obvious alternative to this approach would be to exclude data from subjects who had failed to fulfil learnability criteria on one or more types of task. However, the problem with that approach would be that it might be the data representing the most extreme inter-task differences which would be excluded — arguably the data of greatest importance. For example, say that four of the subjects had failed to fulfil a

learnability criterion with type D tasks, but had fulfilled the criterion for tasks of all other types. If their data were excluded, this would, in effect, mean that the results were being biased in favour of type D tasks. Results are summarised in table 7.9.

Measure	Type A	Type B	Type C	Type D
Trials to last error	1.89 (sd = 1.64)	3.58 (sd = 2.74)	5.14 (sd = 4.90)	6.19 (sd = 3.58)
Trials to time on task < 6 s.	3.69 (sd = 2.81)	4.75 (sd = 2.28)	5.25 (sd = 2.83)	6.06 (sd = 2.91)

Table 7.9. Mean learnability values.

Again the results supported the predictions, except that the difference in learnability between task types C and D did not reach statistical significance (table 7.10). Note that here predictions such as A > C means that it was expected that performance would reach criteria level after fewer trials for type A tasks than with type C tasks.

Comparison	Prediction	Trials to last error	Trials to < 6 s on task
A vs C	A > C	$p < 0.01$, T = 47 (Tc = 137, n = 30)	$p < 0.01$, T = 120, n = 33
B vs D	B > D	$p < 0.01$, T = 69 (Tc = 147, n = 31)	$p < 0.01$, T = 151 , n = 35
A vs B	A > B	$p < 0.01$, T = 76 (Tc = 126, n = 29)	$p < 0.02$, T = 124.5, n = 31
C vs D	C > D	*†NS, T = 223 (Tc = 182, n = 34)*	*†NS, T = 177, n = 32*

Table 7.10. Comparison of learnability values between task types. Where tied values have been excluded, so that n < 36 this is indicated.
† Marks failure of prediction.

7.2.3 EUP

Performance over the last three trials was taken as being indicative of EUP. By this stage subjects were repeating tasks for the tenth, eleventh, and twelfth times, which, it was assumed, represented a reasonable level of experience. From glancing at the performance curves in figures 7.2 and 7.3, it appears that these assumptions are fairly reasonable. However, as an additional check linear regression analyses of time on tasks versus cumulative numbers of each type of task completed, were also performed on a plot of the mean data. This idea was similar to that employed by Jordan and O'Donnell (1992), except that they had checked subjects data individually. Here, the purpose was merely to check that average performance was reasonably steady, so that the chosen criterion was reasonable. Calculated R-squared values of .999, .999, .998, and .997 obtained for task types A, B, C, and D respectively, indicated this to be so.Figures 7.4 and 7.5 illustrate performance over the last few trials of the first experimental session.

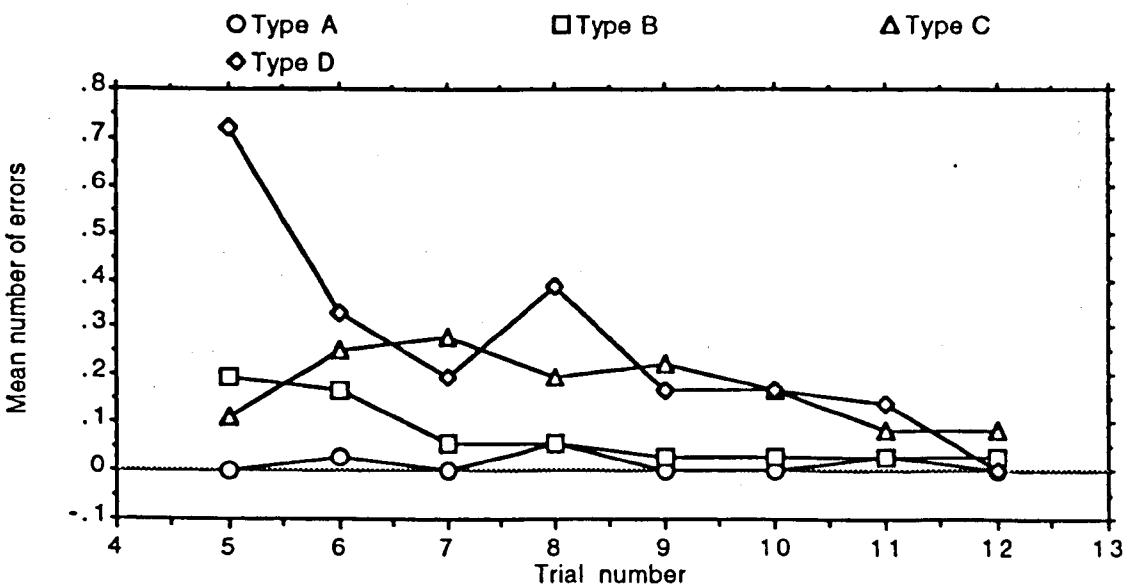


Figure 7.4. Mean number of errors on later trials for each type of task.

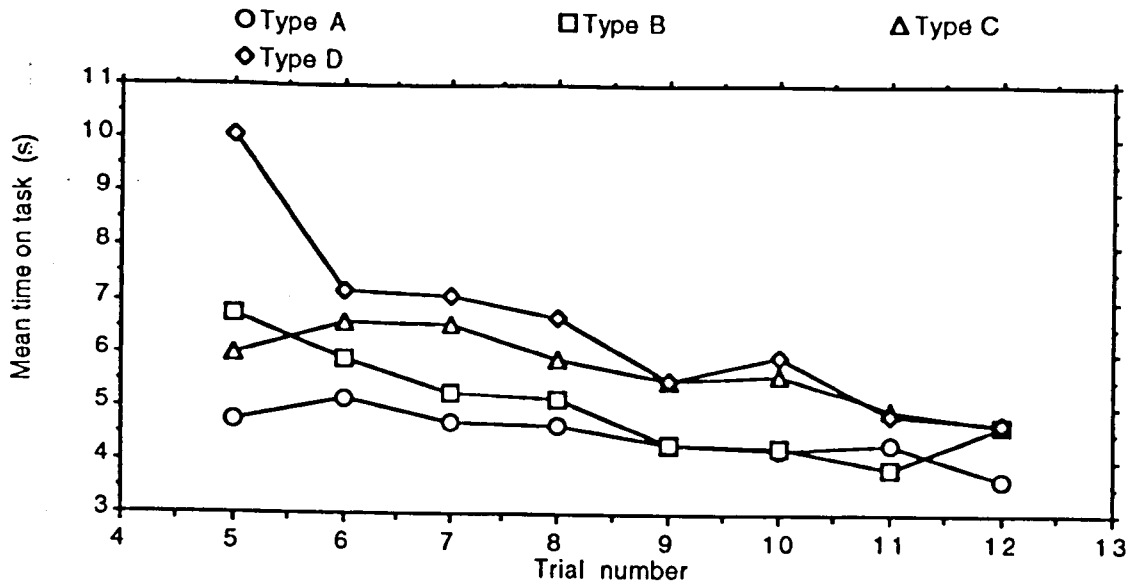


Figure 7.5. Mean time on task on later trials for each type of task.

Table 7.11 gives a summary of EUP values — again, the lower the number of errors and time on task, the higher the EUP.

Measure	Type A	Type B	Type C	Type D
Errors	0.03 (sd = 0.17)	0.08 (sd = 0.28)	0.33 (sd = 0.89)	0.23 (sd = 0.49)
Time	4.0 s (sd = 1.3 s)	4.2 s (sd = 1.0 s)	5.0 s (sd = 2.6 s)	5.3 s (sd = 2.3 s)

Table 7.11. Mean EUP values (Figures quoted for errors are the means of each subjects' *total* number of errors during the last three trials (due to the small numbers of errors committed at this stage), whereas time values are the *mean* time per trial).

The predictions about the effects of consistency on EUP were supported by the results, although the difference between the number of errors committed on task types B and D was not statistically significant.

Comparison	Prediction	Errors	Time on task
A vs C	A > C	p < 0.10*, T = 2.5, n = 7	p < 0.05, T = 192
B vs D	B > D	†*NS, T = 8 n = 10*	p < 0.01, T = 131.5
A vs B	A = B	NS, T = 2.5 (Tc < 0, n= 4)	NS, T = 249.5
C vs D	C = D	NS, T = 26.5 (Tc = 8, n = 10)	NS, T = 254

Table 7.12. Comparison of EUP values between task types. Where tied values have been excluded from the analysis, so that n < 36, this has been indicated. † Indicates failure of prediction.

7.2.4 System potential

Although the action-rules for the tasks set differed in terms of the menus to be opened and the commands to be selected, the general form of each was similar. Also, tasks were balanced for task type across interfaces. Therefore, each type of task should support the same level of system potential both in terms of time and errors (obviously, zero errors would be the minimum possible).

7.2.5 Re-usability

Performance on the trial performed in the second session was taken as being indicative of re-usability. Mean performance values are listed in table 7.13.

Measure	Type A	Type B	Type C	Type D
Errors	0.00 (sd = 0.00)	0.08 (sd = 0.37)	0.06 (sd = 0.23)	0.08 (sd = 0.28)
Time on task	4.48 s (sd = 1.54 s)	5.63 s (sd = 2.83 s)	5.68 s (sd = 2.16 s)	5.62 s (sd = 1.88 s)

Table 7.13. Mean re-usability values.

Again the experimental predictions were checked via WSR tests for statistical significance — results are listed in table 7.14.

Comparison	Prediction	Errors	Time on task
A vs C	A > C	*†NS, T = 0, n = 2*	p < 0.01, T = 148.5
B vs D	B > D	*†NS, T = 5, n = 4*	*†NS, T = 307.5*
A vs B	A > B	*†NS, T = 0, n =2*	p < 0.01, T = 161
C vs D	C > D	*†NS, T = 6, n = 5*	*†NS, T = 320.5*

Table 7.14. Comparison of re-usability values between task types. Where tied values have been excluded from the analysis, so that n < 36, this has been indicated. † Indicates failure of prediction.

The predictions about the effect of consistency on re-usability receive little support from these results. The lack of significance gained from the error data is perhaps not surprising considering the comparative insensitivity of the measure as compared to time on task. However, even the time on task data does not give comprehensive support. Comparison of performance on task type A compared with that on types B and C, does show the predicted effect for both set and rule compatibility. However, this is not mirrored in the comparison between performance on type D tasks, and that on types B and C. Here no significant effects were found for either set or rule compatibility, indeed inspection of the means showed little difference in performance on all three types of task.

This may possibly be due to the experimental design. After all, if subjects had been away from the interface long enough that command placements had been totally forgotten, then surely rule compatibility would have an effect regardless of set compatibility. Similarly, if subjects had returned for the second experimental session only an hour after the first, then set compatibility seems likely be important regardless of rule compatibility.

7.2.6 Overall success

Of the other 24 predictions tested (i.e. excluding those about re-usability) (3 types of usability, 2 measures of each, 4 pair-wise comparisons of conditions), 21 were supported, and 3 were not (identified by † and italics in tables 7.8, 7.10, 7.12). Inspection of the numerical results in those 3 cases, however, shows large (but not statistically significant) differences in the predicted direction. Similarly where the predictions were for no difference, values were not only not significantly

different, but inspection shows them to be very close to each other compared to the other values in the set. The group data therefore strongly supports the predictions, and hence the idea that whilst rule compatibility is more salient an influence on users' early interactions, set compatibility is of more importance later on.

7.2.7 Inter-subject variability

When the results from the four groups of subjects were looked at individually, they were generally in line with the overall results. The one exception to this was the marked difference between the performances of male and female subjects at EUP (see figures 7.6 and 7.7).

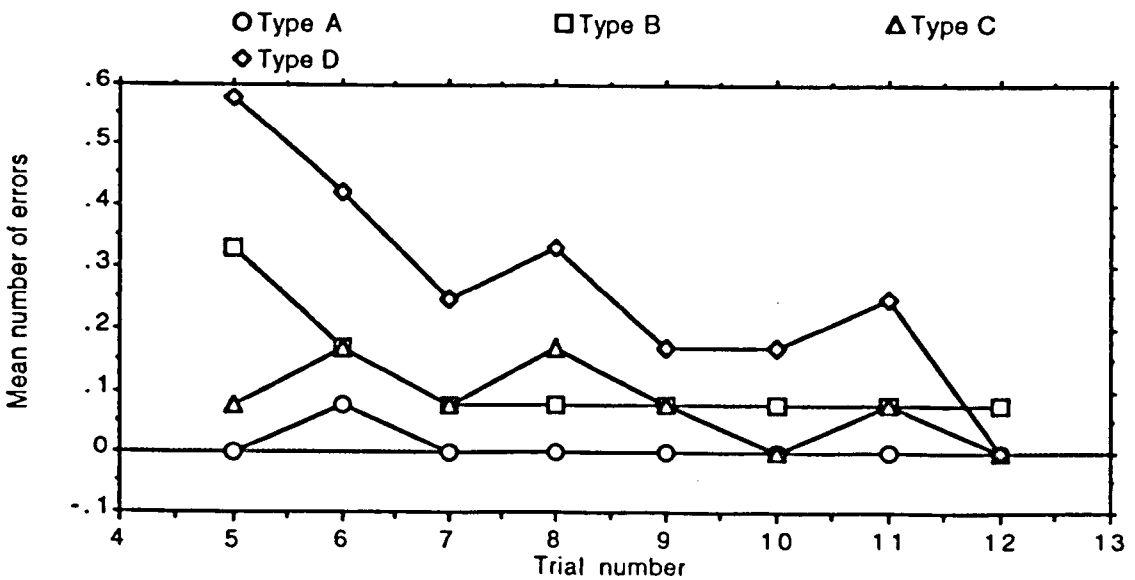


Figure 7.6. Mean number of errors on later trials for each task type (male subjects only).

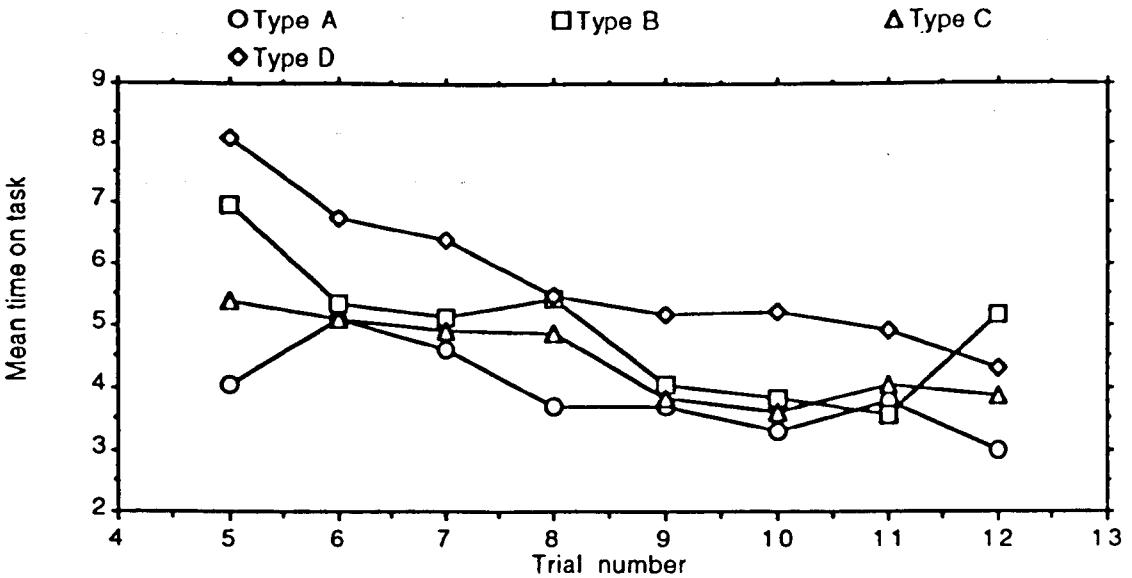


Figure 7.7. Mean time on task on later trials for each task type (male subjects only).

Whilst the direction of inter-task differences in the overall results quoted are representative of performance by both female groups (those with and without previous experience), the results from the two male groups showed different effects (note that there were more female subjects, hence the effect from the male subjects' performance was "swamped" in the combined data). Combining results for the two male groups at EUP, statistically significant effects were found for rule compatibility, but not for set compatibility (see tables 7.15 and 7.16). This result is both contrary to that for female subjects, and contrary to the predictions.

Measure	Type A	Type B	Type C	Type D
Errors	0.00 (sd = 0.00)	0.25 (sd = 0.45)	0.08 (sd = 0.26)	0.42 (sd = 0.70)
Time	3.4 (sd = 0.9)	4.2 (sd = 1.1)	3.8 (sd = 0.7)	4.8 (sd = 1.6)

Table 7.15. Mean EUP values — male subjects only.

Comparison	Prediction	Errors	Time on task
A vs C	A > C	*†NS, T = 1, n = 1*	*†NS, T = 28*
B vs D	B > D	*†NS, T = 5, n = 5*	*†NS, T = 22*
A vs B	A = B	NS, T = 0, n = 3	*†p < 0.05, T = 10*
C vs D	C = D	NS, T = 0, n = 3	*†p < 0.05, T = 13*

Table 7.16. Comparison of EUP values between tasks — male subjects only. Where sample size < 12 (after extracting tied values), this is indicated. † Marks failures of prediction.

No effects were found for level of previous experience with other word processors and the Macintosh operating environment.

7.3 DISCUSSION

The performance of male subjects at EUP was the one result with a strikingly different pattern from the predictions. A look at the data (table 7.15) suggests both (A vs. B, C vs. D) that there is still a distinct effect of rule compatibility (while it was predicted that this would have disappeared at EUP), and (A vs. C, B vs. D) that there is little or no effect of set compatibility (which was predicted to be considerable at EUP).

It might seem possible that this represents a higher degree of learning (or over-learning), comparable to having learned not just the regular part of a language's grammar but all its exceptions too. However inspection of individual data shows that the errors by males were different in kind, rather than generally fewer, than errors by females. In fact the *pattern* of results is the same as it was for first time use (though the score values are much better) — the opposite pattern to that of any "advanced learning" hypothesis. Thus "over-learning" is not a plausible explanation.

The individual data also shows an essentially complete dichotomy between males and females in our sample. It is as if two populations, identified by gender, were doing different things — had different mental strategies.

All the predictions made, both in this thesis and in the consistency literature as a whole, about the effects of consistency on performance rely on the assumption

that users will group "similar" tasks together, and attempt to treat them in similar ways. It may be that, in this case, that although male subjects initially treated commands as pairs (the effect of set compatibility on learnability indicates this), they may have stopped this by the time they reached EUP, and resorted to considering them individually. Or rather, that they abandoned the attempt to group commands together in favour of a strategy of learning separate associations between each command and its menu title. Perhaps the low amount of set compatibility in the interface overall caused this change of learning strategy. It seems to suggest that performance and learning can be affected not only by the design of an interface and how the new knowledge fits with a user's prior knowledge, but also by learning strategies, or perhaps more conscious views about how to conceive the regularities in the design. If these suggestions about male users' strategy are correct, then set compatibility would not be expected to influence EUP for these users, as for them there were no sets.

7.4 CONCLUSIONS

The experimental results largely support the experimental hypotheses, and thus, by corollary, the idea of two distinct types of consistency which will affect usability in different ways. The comprehensive model of usability which was employed in this study gave insights into these differential effects, particularly the way the effects of the two types of consistency change with user learning. Thus it was found that at the earliest stages of learning (guessability) rule compatibility proved more influential, both types of inconsistency affected learnability (although probably for different reasons), set compatibility had the greater influence on EUP, whilst both affected re-usability.

The gender difference found, and the possible interpretation of it in terms of learning strategy, serves to reinforce Reisner's (1990) simple but important point: that it is the user not the investigator who determines the groupings (descriptions) of items. The approach taken here was to choose commands such that the fit between them was, apparently, so obvious that users' groupings would be predictable. The gender result, however, shows that, even in such a seemingly clear cut situation, investigators may not have the power to predict this, and that consistency — how items fit or resist fit with each other — is partly determined by complex individual factors.

7.5 SUMMARY

An experimental study was designed to test the predictions about the effects of set and rule compatibility on the various components of usability. The context of the study — the invocation of menu commands — meant that consistency was concerned with the fit or resistance of command names with other command names, and command names with menu headings. Generally, the results strongly supported the predictions — rule compatibility affecting performance during early interactions, with set compatibility becoming more important later on.

Chapter 8

TESTING THE GUESSABILITY PREDICTIONS

So far, the predictions about the effect of consistency on guessability have only partially been investigated. The guessability measure in the first study was taken from subjects attempting tasks with no previous experience of any similar tasks with that interface. As expected, a large effect was found for rule compatibility. However, another prediction was that, if a user had previous experience of other tasks in the same set with an interface, then set compatibility would have an effect on guessability. The study reported in this chapter was designed to address this issue.

As with the first study, guessability was measured for four separate types of task (see table 8.1).

	RULE COMPATIBLE	RULE INCOMPATIBLE
SET COMPATIBLE	Type A	Type B
SET INCOMPATIBLE	Type C	Type D

Table 8.1. Set and rule compatibility of experimental tasks.

The predictions about the effects of consistency on guessability were as follows (table 8.2).

EXPERIENCE WITH THE INTERFACE FOR SIMILAR TASKS	EFFECT OF SET COMPATIBILITY	EFFECT OF RULE COMPATIBILITY
No previous experience	A = C, B = D	A > B, C > D
High previous experience	A > C, B > D	A = B, C > D*

Table 8.2 Experimental hypotheses for study 2.

The predictions are fairly similar to those about how set and rule compatibility affect usability as a user moves through from guessability to EUP on a task —

rule compatibility being more important in early interactions, with set compatibility mattering more later. Perhaps, then, overall experience with a group of tasks may be just as influential, in terms of how consistency affects usability, as the amount of experience on any particular individual task.

However, there may be one noticeable difference. The difference between a user at EUP on a task and one attempting a task for the first time after previous experience with others in the set, is that the former needs to recall or recognise an action-rule, whilst the latter must infer one. Thus, although it was predicted that set compatibility would be the main determinant of guessability for previously experienced users, rule compatibility was thought to be important where set compatibility was absent (see prediction marked *). This was because it was anticipated that users would initially try to generalise schema from their previous experiences of using an interface, but having found that this did not provide them with the required rule might resort to using their world knowledge. A rule compatible task, then, may at least be guessable at the second attempt. However, if the task was neither set nor rule compatible, then reverting to world knowledge would still not provide the user with the correct schema.

8.1 THE STUDY

8.1.1 Apparatus

As for the previous study Microsoft Word 4 for the Macintosh was specially adapted for the experiment. A total of 20 commands were distributed across four menus. The commands were chosen so that (it was assumed) users would divide them into four groups of five "similar" commands. The invocation of one command per menu was selected as the experimental task, and the activation of the other four were filler tasks.

Four separate interfaces were created. For each interface all the experimental tasks were to be of a particular type (analysis was to be between groups). Figures 8.1 and 8.2 illustrate two such interfaces (the others are illustrated in appendices 8.1 and 8.2). Commands whose invocation was to be an experimental task are shown in italics.

FORMAT	FONT	DOCUMENT	WINDOW
COPY FORMATS	LARGER FONT SIZE	MOVE TO END DOCUMENT	ZOOM WINDOW
PLAIN FORMAT	SHAPES FONT	PAGE DOCUMENT	SPLIT WINDOW
UNDERLINE FORMAT	*SMALLER FONT SIZE*	SCROLL DOCUMENT	TOP WINDOW
DEFINE FORMATS	TYPE FONT	*MOVE START DOCUMENT*	OPEN FOOTNOTE WINDOW
FIND FORMATS	CHANGE FONTS	SELECT WHOLE DOCUMENT	*NEW WINDOW*

Figure 8.1. An experimental menu configuration (configuration A).

FORMAT	FONT	DOCUMENT	WINDOW
LARGER FONT SIZE	ZOOM WINDOW	PLAIN FORMAT	*COPY FORMATS*
SHAPES FONT	SPLIT WINDOW	UNDERLINE FORMAT	MOVE TO END DOCUMENT
TYPE FONT	TOP WINDOW	*SMALLER FONT SIZE*	PAGE DOCUMENT
CHANGE FONTS	*MOVE TO START DOCUMENT*	DEFINE FORMATS	SCROLL DOCUMENT
NEW WINDOW	OPEN FOOTNOTE WINDOW	FIND FORMATS	SELECT WHOLE DOCUMENT

Figure 8.2. An experimental menu configuration (configuration D).

With configuration A the invoking of the experimental commands would be both set and rule compatible (type A), whilst for configuration D they would be neither set nor rule compatible (type D). Formalisations of the task-action rules associated with each interface are given in table 8.3 and 8.4. Note, that as in the previous interface, three menus were left empty — their headings acting as potential distractors.

Taskset
TASKSET = [*invoke "COPY FORMATS"*, invoke "PLAIN FORMAT", invoke "UNDERLINE FORMAT", invoke "DEFINE FORMATS", invoke "FIND FORMATS", invoke "LARGER FONT SIZE", invoke "SHAPES FONT", *invoke "SMALLER FONT SIZE"*, invoke "TYPE FONT", invoke "CHANGE FONTS", invoke "MOVE TO END OF DOCUMENT", invoke "PAGE DOCUMENT", invoke "SCROLL DOCUMENT", *invoke "MOVE TO START DOCUMENT"*, invoke "SELECT WHOLE DOCUMENT", invoke "ZOOM WINDOW", invoke "SPLIT WINDOW", invoke "TOP WINDOW", invoke "OPEN FOOTNOTE WINDOW", *invoke "NEW WINDOW"*]

Schema sets (agent = designer)
SET W = [*COPY FORMATS*, PLAIN FORMAT, UNDERLINE FORMAT, DEFINE FORMATS, FIND FORMATS]

SET X = [LARGER FONT SIZE, SHAPES FONT, *SMALLER FONT SIZE*, TYPE FONT, CHANGE FONTS]

SET Y = [MOVE TO END OF DOCUMENT, PAGE DOCUMENT, SCROLL DOCUMENT, *MOVE TO START DOCUMENT*, SELECT WHOLE DOCUMENT]

SET Z = [ZOOM WINDOW, SPLIT WINDOW, TOP WINDOW, OPEN FOOTNOTE WINDOW, *NEW WINDOW*]

Rule schemata (agent = designer)
to invoke W -> Open FORMAT menu + select W
to invoke X -> Open FONT menu + select X
to invoke Y -> Open DOCUMENT menu + select Y
to invoke Z -> Open WINDOW menu + select Z

Schema sets (agent = user)
SET P = [*COPY FORMATS*, PLAIN FORMAT, UNDERLINE FORMAT, DEFINE FORMATS, FIND FORMATS]

SET Q = [LARGER FONT SIZE, SHAPES FONT, *SMALLER FONT SIZE*, TYPE FONT, CHANGE FONTS]

SET R = [MOVE TO END OF DOCUMENT, PAGE DOCUMENT, SCROLL DOCUMENT, *MOVE TO START DOCUMENT*, SELECT WHOLE DOCUMENT]

SET S = [ZOOM WINDOW, SPLIT WINDOW, TOP WINDOW, OPEN FOOTNOTE WINDOW, *NEW WINDOW*]

Rule schemata (agent = user)
to invoke P -> Open FORMAT menu + select P
to invoke Q -> Open FONT menu + select Q
to invoke R -> Open DOCUMENT menu + select R
to invoke S -> Open WINDOW menu + select S

Set compatibility
P matches W
Q matches X
R matches Y
S matches Z

Rule compatibility
Rules for all tasks match

Table 8.3. Formalisation of task-action rules associated with configuration A. Experimental tasks are shown in italics.

Taskset
TASKSET = [*invoke "COPY FORMATS"*, invoke "PLAIN FORMAT", invoke "UNDERLINE FORMAT", invoke "DEFINE FORMATS", invoke "FIND FORMATS", invoke "LARGER FONT SIZE", invoke "SHAPES FONT", *invoke "SMALLER FONT SIZE"*, invoke "TYPE FONT", invoke "CHANGE FONTS", invoke "MOVE TO END OF DOCUMENT", invoke "PAGE DOCUMENT", invoke "SCROLL DOCUMENT", *invoke "MOVE TO START DOCUMENT"*, invoke "SELECT WHOLE DOCUMENT", invoke "ZOOM WINDOW", invoke "SPLIT WINDOW", invoke "TOP WINDOW", invoke "OPEN FOOTNOTE WINDOW", *invoke "NEW WINDOW"*]

Schema sets (agent = designer)
SET W = [LARGER FONT SIZE, SHAPES FONT, TYPE FONT, CHANGE FONTS, *NEW WINDOW*]

SET X = [ZOOM WINDOW, SPLIT WINDOW, TOP WINDOW, *MOVE TO START OF DOCUMENT*, OPEN FOOTNOTE WINDOW]

SET Y = [PLAIN FORMAT, UNDERLINE FORMAT, *SMALLER FONT SIZE*, DEFINE FORMATS, FIND FORMATS]

SET Z = [*COPY FORMATS*, MOVE TO END OF DOCUMENT, PAGE DOCUMENT, SCROLL DOCUMENT, SELECT WHOLE DOCUMENT]

Rule schemata (agent = designer)
to invoke W -> Open FORMAT menu + select W
to invoke X -> Open FONT menu + select X
to invoke Y -> Open DOCUMENT menu + select Y
to invoke Z -> Open WINDOW menu + select Z

Schema sets (agent = user)
SET P = [*COPY FORMATS*, PLAIN FORMAT, UNDERLINE FORMAT, DEFINE FORMATS, INVOKE FIND FORMATS]

SET Q = [LARGER FONT SIZE, SHAPES FONT, *SMALLER FONT SIZE*, TYPE FONT, CHANGE FONTS]

SET R = [MOVE TO END OF DOCUMENT, PAGE DOCUMENT, SCROLL DOCUMENT, *MOVE TO START DOCUMENT*, SELECT WHOLE DOCUMENT]

SET S = [ZOOM WINDOW, SPLIT WINDOW, TOP WINDOW, OPEN FOOTNOTE WINDOW, *NEW WINDOW*]

Rule schemata (agent = user)
to invoke P -> Open FORMAT menu + select P
to invoke Q -> Open FONT menu + select Q
to invoke R -> Open DOCUMENT menu + select R
to invoke S-> Open WINDOW menu + select S

Set compatibility
No sets match

Rule compatibility
Rules for no tasks match

Table 8.4. Formalisation of task-action rules associated with configuration D. Experimental tasks are shown in italics.

The other two menu configurations supported experimental tasks of types B and C respectively (see appendix 8.3 and 8.4 for formal descriptions).

8.1.2 Subjects

Thirty-two Glasgow University students participated in the study, 16 males and 16 females. 15 had used a Macintosh computer before, and 24 had word processing experience, whilst 13 had word processing experience on a Macintosh. Allocation of subjects to groups was balanced for gender, experience with Macintosh computers, and experience with word processors. As the interfaces were designed especially for the experiment, no subject had knowledge of the application program.

8.1.3 Experimental design

Each subject performed four blocks of filler tasks, each followed by an experimental task. A block of fillers consisted of invoking every filler command on the interface in an order randomised by "pulling numbers out of a hat". This meant, then, that each subject would perform a total of 68 tasks (16 fillers four times each, plus four experimental tasks once each) in the following sequence: sixteen fillers, experimental task 1, sixteen fillers, experimental task 2, sixteen fillers, experimental task 3, sixteen fillers, experimental task 4. Note, that of the sixteen fillers only four were connected with each experimental task. However, the design included repetitions of all sixteen, as it was felt that completion of a variety of different types of task might be more representative of a realistic working situation. After all, with real word processing tasks, users may not work in blocks of similar tasks, rather it seems more likely that they would intersperse tasks of different types. The order in which the experimental tasks were presented was balanced between subjects within each of the interfaces via Latin squares (so, for example, if one of the subjects using configuration A had "invoke Copy formats" as experimental task 1, another might have, say, "invoke New window").

Performance was measured in terms of number of errors made and time on task. As for the previous study, an error was classified as opening an inappropriate menu.

8.1.4 Procedure

After reading an introduction to the experiment (appendix 8.5) and completing a personal details interview, subjects were shown how to invoke a command. This was done by pressing the enter key on the number key-pad, typing a number from the main key-pad to open a menu, moving up and down the menu using cursor keys, and pressing return when the appropriate command was highlighted. This method of invocation differed from study 1 only in that the enter key was used to access the menus rather than a combination of the "apple" and "tab" keys. Subjects in the previous study had sometimes made slips at this stage in the task — a source of (minor) experimental noise which was thus eliminated for this study.

As for the previous study, subjects practised command invocation using a made up menu, not for use during the experiment, so that they had no experience of the experimental interface prior to the start of the experimental session. The experimenter set tasks by reading out the name of the command to be invoked. As soon as one task was completed the next was set. A text file was displayed on the screen, which remained throughout the experimental session. As with the first study, it was not expected that subjects would have a clear idea of the functionality of the commands which they were asked to invoke. It was assumed, then, that subjects' task groupings, and their action-rule expectations would be made on the basis of command names and menu names. Time on task and errors were recorded, and the experimenter followed a written protocol (appendix 8.6). Each subject's experimental session lasted about half an hour.

8.2 RESULTS AND DISCUSSION

Mean time on task and errors for the experimental tasks are listed in tables 8.5 and 8.6 respectively.

Blocks of fillers completed	TYPE A	TYPE B	TYPE C	TYPE D
1	5.4 s (sd = 1.4 s)	17.6 s (sd = 6.0 s)	21.0 s (sd = 9.4 s)	14.8 s(sd = 6.8 s)
2	5.0 s (sd = 0.8 s)	8.8 s (sd = 6.7 s)	18.4 s (sd = 8.8 s)	19.8 s (sd = 6.4 s)
3	5.2 s (sd = 3.3 s)	6.4 s (sd = 1.8 s)	16.5 s (sd = 13.8 s)	16.4 s (sd = 6.4 s)
4	4.8 s (sd = 1.2 s)	7.4 s (sd = 3.8 s)	15.8 s (sd = 6.2 s)	11.8 s (sd = 5.5 s)

Table 8.5. Mean time on experimental tasks.

Blocks of fillers completed	TYPE A	TYPE B	TYPE C	TYPE D
1	0.0 (sd = 0.0)	1.5 (sd = 1.5)	1.4 (sd = 0.9)	1.3 (sd = 1.3)
2	0.0 (sd = 0.0)	0.4 (sd = 0.4)	1.6 (sd = 1.1)	2.0 (sd = 0.9)
3	0.0 (sd = 0.0)	0.1 (sd = 0.4)	1.4 (sd = 1.5)	1.8 (sd = 1.0)
4	0.0 (sd = 0.0)	0.4 (sd = 0.5)	1.5 (sd = 0.8)	1.1 (sd = 1.0)

Table 8.6. Mean number of errors made on experimental tasks.

The experimental predictions were checked by carrying out Mann-Whitney "U" tests to check for statistically significant differences between results. Tables 8.7 and 8.8 show the results of these tests. The predictions listed are those relating the effects of set and rule compatibility on guessability for those with a high level of experience at the interface with similar tasks. Therefore, performance on the task set after four blocks of fillers, is the most suitable test of these.

Comparison	Prediction	After 1 block of fillers	After 2 blocks of fillers	After 3 blocks of fillers	After 4 blocks of fillers
A vs C	A > C	0 (p = 0.00)	1 (p = 0.00)	4 (p = 0.00)	0 (p = 0.00)
B vs D	B > D	21 (NS)	8 (p = 0.01)	1 (p = 0.00)	16.5 (NS)
A vs B	A = B	0 (p = 0.00)	23 (NS)	16 (p = 0.05)	16.5 (NS)
C vs D	C > D	23 (NS)	29.5 (NS)	25 (NS)	15 (p = 0.08)*

Table 8.7. Comparison of guessability (time on task) between task types. Numbers represent "U" values from Mann-Whitney non-parametric test, significance levels are given in brackets. (* Indicates marginal significance).

Comparison	Prediction	After 1 block of fillers	After 2 blocks of fillers	After 3 blocks of fillers	After 4 blocks of fillers
A vs C	A > C	0 (p = 0.00)	4 (p = 0.00)	4 (p = 0.00)	0 (p = 0.00)
B vs D	B > D	29 (NS)	5.5 (p = 0.00)	5.5 (p = 0.00)	17 (NS)
A vs B	A = B	4 (p = 0.00)	24 (NS)	28 (NS)	20 (NS)
C vs D	C > D	28.5 (NS)	25.5 (NS)	20.5 (NS)	23.5 (NS)

Table 8.8. Comparison of guessability (errors) between task types. Numbers represent "U" values from Mann-Whitney non-parametric test, significance levels are given in brackets. (* Indicates marginal significance).

Considering performance after four blocks of fillers, the outcomes of the Mann-Whitney U tests only support two of the predictions in terms of both time on task and errors (A > C, and A = B). However, the prediction that type B tasks will be more guessable than those of type D was strongly supported by tests done on data taken after two and three blocks of fillers, and from the mean results after four blocks (type D tasks took, on average, over 60% longer, and subjects made almost three times as many errors). Conversely, the prediction that type C tasks would be more guessable than type D gained no support. If anything subjects performed type D tasks more quickly throughout the session — indeed after four filler blocks a marginally significant advantage is shown for these in terms of time.

Figures 8.3 and 8.4 summarise the data, showing how the influences of set and rule compatibility on guessability change with the user's level of prior experience with an interface. The first points on the plots are taken from the previous study (reported in chapter 7).[2] Overall, the effects were as expected, rule compatibility proving more important for users with little prior experience of the interface, with set compatibility having the larger effect as experience increases.

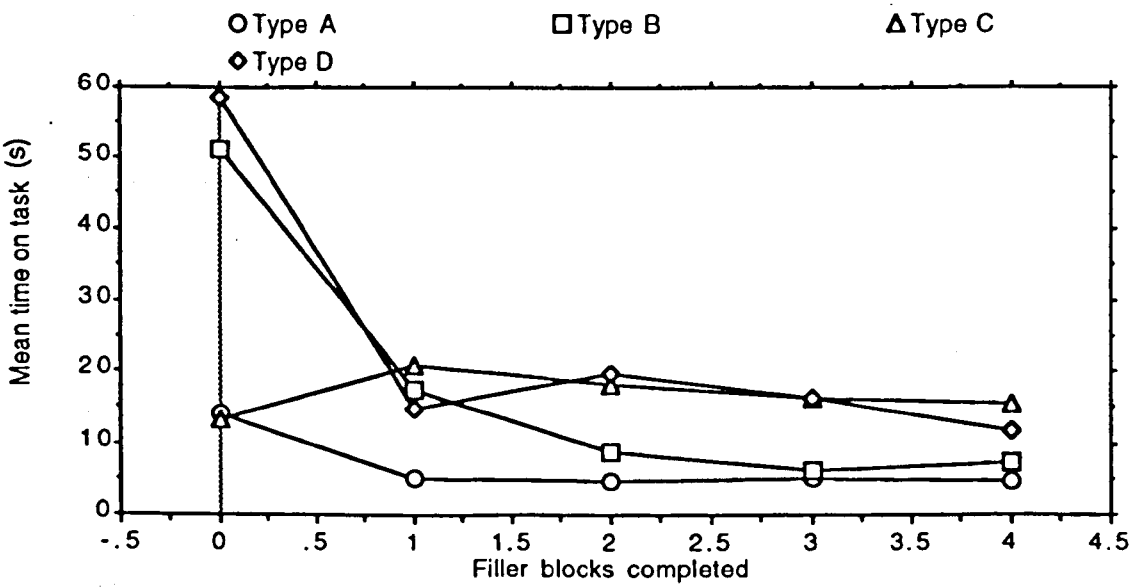


Figure 8.3. Mean time on experimental tasks (seconds) versus previous experience with similar tasks (in terms of blocks of fillers previously completed) for each task type.

[2]Although conditions and experimental design were not identical (for example, there were less menu contents to search through, and the mechanism for command activation was slightly different) the tasks in the previous study were fairly similar to those set here. Thus, this data may be useful for purposes of comparison.

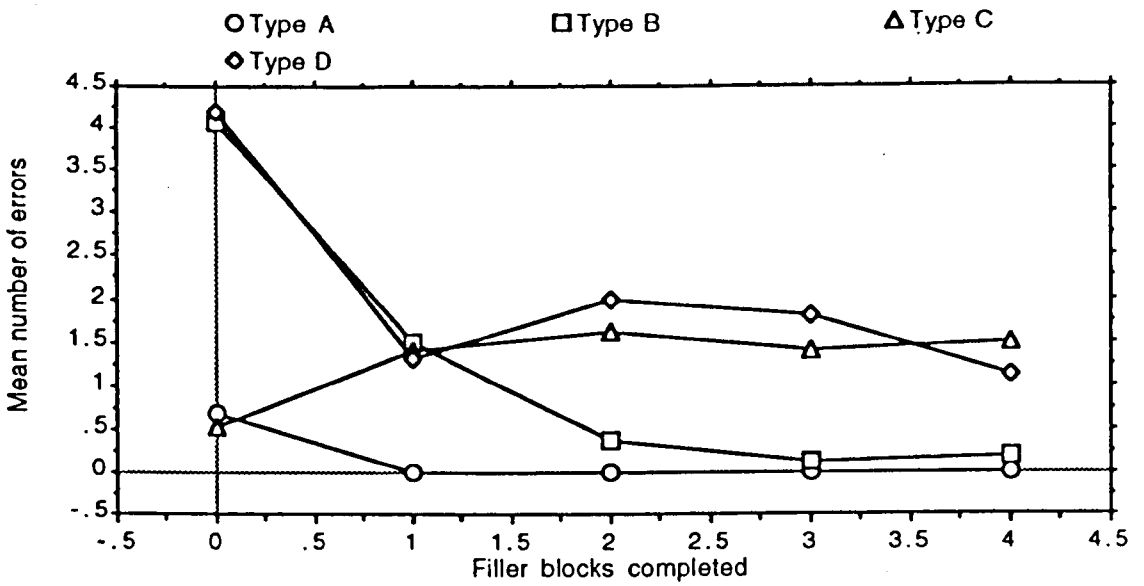


Figure 8.4. Mean number of errors on experimental tasks versus experience with similar tasks (in terms of blocks of fillers previously completed) for each task type.

8.2.1 Comparison to guessability, learnability and EUP

It was predicted that the results of this study would, to an extent, mirror those of the previous one, and this proved, generally, to be the case. Rule compatibility had the larger effect at the earlier stages of interaction, and set compatibility was the more salient factor as experience with a set of tasks increased. In particular set compatibility can make a new task guessable for a user with previous experience of similar tasks, even if its associated action-rule was not originally predictable. Perhaps, then, contrary to the emphasis on the need to distinguish between guessability, learnability and EUP, where there is set compatibility, a user's performance on a task would be more accurately predicted from total experience with a schema set, rather than the number of repetitions of any individual task.

However, in situations where there is no set compatibility, the issues will be different. A user at EUP on a task which is not set compatible, is likely to suffer from interference from schemata associated with other tasks (in the user's set). Thus, he or she may be choosing between known schema, possibly leading to

additional mental processing time, and residual errors. However, when attempting to perform a new task, the appropriate schema may not even be part of the users' mental contents. If set compatibility is absent, trying to generalise from other tasks will not work. A user will, then, have to discover the schema, either through exploration of the interface, or by inference from world knowledge. So, whilst performance on a set incompatible task might be expected to improve with repetition (as in previous study), users' first attempts at this task will not be aided by any amount of prior experience of other tasks in the same set. Figures 8.3 and 8.4, confirm this (although users did get slightly faster on task types C and D this was probably due to becoming more practiced with the mechanics of command activation).

8.2.2 Discovering a schema in the absence of set compatibility

One of the predictions investigated was that in the absence of set compatibility, rule compatibility would aid guessability (C > D). This was because it was thought that, having failed to generalise a schema from other tasks, users could then revert to calling on world knowledge to infer a rule, rather than exploring the interface on a trial and error basis. However, so far the experimental results do not provide any evidence for this claim.

It is possible that the design of the experiment was not conducive to highlighting any effects which did exist. The "problem space" in the study was comparatively small — with only four menus containing commands — so subjects may have felt that after their initial attempts to find a command had failed, they might as well just check the menus for its position. The artificiality of the experiment may also have contributed to the disguising of any effects. In both of the set incompatible interfaces the filler commands were all placed on rule incompatible menus, making the interfaces as a whole overwhelmingly rule incompatible. Thus, users may have ruled out the strategy of referring to world knowledge believing it to be un-helpful in this experimental situation. Clearly, in "real life" contexts it is rare that an interface would be so wholly or obviously rule incompatible.

However, a post-hoc investigation did find some support for the prediction. Data from the subjects using interface configurations C and D was analysed for instances of successful task completions at the second attempt. It was assumed that if users reverted to world knowledge, in the absence of set compatibility, then

they would be likely to get type C tasks right at the second attempt, but not necessarily type D tasks. Results are summarised in table 8.9.

Blocks of fillers completed	Type C	Type D
1	4	2
2	3	3
3	6	2
4	5	4

Table 8.9. Number of users successfully completing tasks at the second attempt (n = 8, for each condition).

A between subjects analysis was also carried out to check for statistically significant differences. This was done by totalling the number of tasks which each subject got right at the second attempt, across the four experimental trials. So, each subject could score a maximum of four and a minimum of zero with respect to this measure. Subjects performing type C tasks got a mean of 2.25 (sd = 0.71) right second time, compared with 1.38 (sd = 0.92) for those doing task type D. This difference proved marginally statistically significant (Mann-Whitney "U" = 14.5, p = 0.07).

8.2.3 Inter-subject variability

In the previous study, investigation of individual subject's data suggested that not all were grouping tasks into sets, but rather that some appeared to be learning task action-rule associations individually. The data from this study was therefore looked at for similar patterns. If there were any subjects who were not grouping tasks, then when performing new type B tasks they would be expected to make errors, regardless of the number of filler tasks previously completed. Meanwhile, those performing type C tasks should get the experimental tasks right first time, regardless of the number of fillers previously completed. (It would not be possible to infer anything from subjects performing type A or D tasks, as here the presence or absence of set and rule compatibility confound each other).

Only three subjects made errors on their final type B tasks (and then only a single error), and for one of these this was a considerable improvement over her performance on the first experimental task (where she made five errors). All of the

subjects attempting type C tasks made errors throughout the experimental session. These results suggest that, at least, the vast majority of subjects were grouping tasks.

A possible reason for this apparent uniformity of subject strategy may, again, be the structure of the experimental interfaces. The design of the interfaces for the study reported in the previous chapter meant that of the eight commands set (twelve times each) only four were set compatible. Therefore, although the strategy of grouping should prove beneficial for half of the activations, it would not prove beneficial for the other half. Furthermore, because the placement of three of the commands gave rule compatibility, if the subject were to simply try and remember individual task-action mappings, then drawing on world knowledge should prove helpful for these commands. Conversely, of the total of 68 commands that subjects were set in this experiment (four repetitions of 16 fillers, plus 4 experimental tasks), at least 64 would be wholly or partly set compatible for every interface (68 on types A and B). However, with the exception of interface configuration A (where every task was set and rule compatible), very few tasks set were rule compatible (none for B, 4 for C, and none for D). Hence, even on interface configuration C the strategy of grouping tasks should prove helpful for 64 out of the 68 tasks, whilst a subject who decided to make inferences from world knowledge would find this strategy successful on only four occasions. Thus, the nature of the interfaces design has dictated that only one strategy can give a reasonable return of success.

8.3 SUMMARY

An investigation of the effects of set and rule compatibility on guessability, for users with previous experience of similar tasks on an interface has been carried out. As predicted, set compatibility had a greater influence on performance.

All of the predictions outlined in chapter 6, encompassing twenty inter-task comparisons (four inter-task comparisons for each five components of usability), have now been tested, in the context of menu driven word processing packages, either in this study, or the one reported in the previous chapter. Statistical support was found (either in terms of time on task, or in terms of errors) for all but three of the predictions. The failure of two of the re-usability predictions was discussed in the previous chapter. The other failure of prediction concerned the effect of rule compatibility on learnability, yet, even here, in terms of mean values performance differences, both in terms of time on task and errors were in the predicted direction.

Chapter 9

APPLYING THE THEORY TO AN ICONIC INTERFACE

Results of the studies reported in the two previous chapters appear to support the predictions about the effect of consistency on usability. Both of these studies concerned command invocation with menu driven interfaces, where both commands and menus had textual display names. It was hoped, however, that the predictions would generalise to other types of interface. The study reported in this chapter was designed to address this question, in the context of using an iconic interface, where commands were represented pictorially, without textual labels.

Again, usability was investigated for four types of task (see table 9.1)

	RULE COMPATIBLE	RULE INCOMPATIBLE
SET COMPATIBLE	Type A	Type B
SET INCOMPATIBLE	Type C	Type D

Table 9.1. Set and rule compatibility of experimental tasks.

This time the experimental hypotheses encompassed all the predictions discussed in chapter 6, with the exception of those relating to re-usability (see table 9.2).

COMPONENT OF USABILITY	EFFECT OF SET COMPATIBILITY	EFFECT OF RULE COMPATIBILITY
Guessability (No previous experience)	A = C, B = D	A > B, C > D
Guessability (Previous experience)	A > C, B > D	A = B, C > D
Learnability	A > C, B > D	A > B, C > D
EUP	A > C, B > D	A = B, C = D
System potential	A = C, B = D	A = B, C = D

Table 9.2. Experimental hypotheses. A > B, for example, is a prediction that type A tasks will be performed better than type B — i.e. at less cost to the user.

Subjects were asked to perform "tasks" using a mock interface to a "file manipulation package". To do this they had to click the mouse first on an object icon representing the file to be manipulated, and then on an operation icon representing the manipulation to be carried out. Object icons were labelled textually — in effect they were simply named buttons — whilst the operation icons had pictorial labels. Consistency, in the context of this study, was manipulated via the labelling and usage of the operation icons.

Set compatibility was connected with whether or not the same operation icon could be used for a particular type of task, regardless of the object being operated on, or whether the operation icon required was object dependent. For example, consider two potential tasks: saving a text file, and saving a statistics file. Imagine that the rules for these were:

to save a text file -> position cursor over TEXT FILE icon + click mouse + position cursor over ICON 1 + click mouse

and:

to save a statistics file -> position cursor over TEXT FILE icon + click mouse + position cursor over ICON 1 + click mouse.

These tasks would be set compatible, as the action-rules for both could be represented by the single schema:

to save SET X -> position cursor over SET X icon + click mouse + position cursor over ICON 1 + click mouse (where SET X = [text file, statistic file]).

However had the rule for saving a statistics file been:

to save a statistics file -> position cursor over STATISTICS FILE icon + click mouse + position cursor over ICON 2 + click mouse

then the tasks would not be set compatible, as separate schemata would be needed to account for the use of different operation icons. Possible versions of "saving" icons are illustrated in figures 9.1 and 9.2.

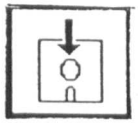


Figure 9.1. Possible label for saving icon.

Figure 9.2. Possible label for saving icon.

Rule compatibility, in this context, is concerned with the a priori suitability of an icon for the operation which it represents. So, in the previous example, if ICON 1 was one which users were inclined to strongly associate with saving, then the actions for saving a text file would be rule compatible. Conversely, if users did not associate ICON 1 with deleting, or if the interface contained another icon which users more strongly associated with this function, then deleting a text file would be a rule incompatible task.

9.1 THE STUDY

9.1.1 Apparatus

Four interfaces were created using Hypercard to run on the apple Macintosh PC. Each contained a total of sixteen object icons (each representing a different type of file), and eight operation icons. These operation icons could be used for a total of four different types of manipulation (there were two operation icons for each): saving files to disk, copying files, deleting files and printing files. The object icons had textual labels, naming the type of file represented as they would be referred to in the experimenter's instructions to subjects: "text file", "statistics file" etc.. Operation icons, meanwhile, had pictorial labels. Each experimental task was performed by first clicking on an object icon, and then on the appropriate operation icon. If a user clicked on a valid combination of icons, feedback would appear in the form of a textual message informing them of the manipulation performed. However, if invalid inputs had been made, a message would appear informing users of this and suggesting that they try again. The interface was programmed such that only combinations required for tasks set in the experiment would be treated as valid. This meant that for each object icon there was only one operation icon which it was appropriate to click on.

Each of the interfaces was designed to support tasks of types A, B, C, and D, so that a within subjects comparison could be used. The interfaces were designed so that experimental tasks were balanced for task type across interfaces. So, if for one interface saving a file were, say, a type A task, then on another it might be a type C task. Balancing was via a Latin square. The spatial layout of the icons was determined at random for each of the four interfaces.

The experiment was designed so that each type of manipulation could be performed on four types of file. Depending upon whether or not tasks were to be set compatible, it would be necessary to use a total of either one or two operation icons in order to carry out a particular manipulation on all four file types. For example, if the icon in figure 9.3 were used in printing references files, graphics files, data files, and programming files, then the interface would be set compatible for this group of tasks. However, if this icon could only be used for printing references files, and the icon in figure 9.4 were needed for printing graphics, data, and programming files then there would be a set incompatibility. Half of the tasks set with each interface would be set compatible — so two of the four types of manipulation set would require just one operation icon each, whilst the other two would require two each. This meant that a total of only six icons per interface would be required for the tasks set, the other two merely acting as potential distractors — necessary in order to manipulate rule compatibility.

Figure 9.3. Possible label for printing icon.

Figure 9.4. Possible label for printing icon.

Rule compatibility was manipulated according to the suitability of the pictorial symbols on the operation icons. A task would be rule compatible if the operation icon required was aptly labelled. However, if the interface contained another icon more suggestive of the operation in question than the icon actually required, then the task would be rule incompatible. The basis of the claims made about the suitability of the pictorial symbols is work done by Maissel (1990).

Maissel performed a study in which he asked subjects to rank groups of symbols. These groups were generated by copying from existing interfaces and previous icon studies. Each of the symbols in a group had been used to represent the same object or operation. So, for example, there would be one group full of symbols representing the operation "delete" and another full of symbols representing the object "file". For the purposes of the consistency study reported here, it is the operation symbols which were of interest.

First, Maissel's subjects were asked to rate each symbol as either "highly appropriate", "O.K.", or "highly inappropriate", for the object or operation which they were designed to represent. Subjects were told that this judgement should be based on three criteria: whether they thought users would be able to understand the meaning of the icon, whether an icon would be easily recognisable, and whether they thought the icon could be confused with one for another operation.

In the second part of the study Maissel asked another set of subjects to rank icons in order of appropriateness. The icons used here were those that performed best and worst in the first evaluation, plus some new designs. The third and final part of the study was a comprehension test — only those icons which had performed particularly well in the earlier parts of the study were used here.

The symbols used for the icons in the experimental interface were ones which had performed either particularly well or particularly badly in Maissel's study. Where an experimental task was to be rule compatible an icon with a suitable label would be used, but where the task was to be rule incompatible an unsuitable icon would be required. The other member of the pair for each operation would be used for filler tasks, or would simply be put into the interface as a distractor, depending on whether the experimental task was of type A, B, C or D. So, for example, type A experimental tasks would require use of a suitable icon, as would its fillers, however an unsuitable icon for the same type of operation would be included in the interface as a potential distractor. Type B experimental tasks would all require the use of an unsuitable icon, but a suitable icon for the same operation would be included in the interface as a distractor. Type C experimental tasks, would require the use of a suitable icon, with an unsuitable icon being used for the fillers, whilst type D experimental tasks would require an unsuitable icon, with a suitable icon being used for the fillers.

The icons chosen for the experimental interface are illustrated in table 9.3, along with their associated scores from Maissel's study. In the first part of his study Maissel assigned a score of +1 to an icon for each subject who rated it as being "highly appropriate" and a score of -1 for each subject who thought it "highly inappropriate", with no score being given where the rating was "O.K.". As there were fourteen subjects, the best possible overall rating an icon could gain was +14, and the worst -14. The scoring for the second part of the study was more complex, involving taking the sum of ranks. For the sake of clarity, what is given in the table below is the final overall position of each icon in the "order of merit" and the number of icons that subjects compared it to.

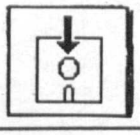
OPERATION	SUITABLE ICON	UNSUITABLE ICON
Deleting		
	Rating +2 Rank 1st / 15	Rating -8 Rank 15th / 15
Copying		
	Rating +1 Rank 1st / 15	Rating -12 Rank 15th / 15
Saving to disk		
	Rating +6 Rank 1st / 15	Rating -13 Rank 15th / 15
Printing		
	Rating +12 Rank 1st / 13	Rating -14 Rank =12th / 13

Table 9.3. Icons chosen for experimental interface.

Unfortunately, a design error was included in each interface — this was similar across all four. For type D tasks (set incompatible and rule incompatible) the intention was that an unsuitable icon be required for the experimental tasks, with a suitable one being used for the fillers. However, due to errors (which were entirely the fault of the author) in the instructions given to the interface

programmer[3], filler tasks were also performed using unsuitable icons, although these were still different from the icons used for the experimental tasks. Unfortunately, this error was not noticed until after the study had been completed. These filler icons are illustrated in table 9.4, along with their associated scores.

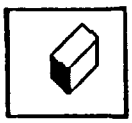
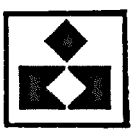
	
Deleting **Rating -4, Rank 13th / 15**	**Copying** **Rating -3, Rank 12th / 15**
	
Saving to disk **Rating -11, Rank 14th / 15**	**Printing** **Rating -12, Rank = 12th / 13**

Table 9.4. Accidentally included filler icons.

This error was not "fatal" — these experimental tasks were still comparatively rule incompatible as they relied on unsuitable icons. However, the rule incompatibility was not as strong as it might have been had highly suitable icons been used for the fillers, as opposed to ones which were only slightly more suitable (according to Maissel) than those used for the experimental tasks. After all, with display based interfaces, the a priori naturalness of using a particular display item may be wholly or partly dependent on what other items are displayed.

The potential consequence of the error, then, was that, in situations where rule compatibility can effect usability (during earlier interactions), performance on type D tasks might not be as bad as would otherwise have been expected. However, the error should not effect performance on any of the other types of experimental task.

Table 9.5 shows a formalisation of the task-action rules associated with one of the four interfaces. The experimental tasks are shown in italics.

---

[3] The programmer, a computer science postgraduate studying icon design, also ran half of the experimental subjects, as aspects of the study were also of interest in the context of her own work. Note, that in all other studies reported all work, including the building of experimental interfaces, was solely the work of the author.

Taskset
TASKSET = [*copy a drawing file*, copy a news file, copy a mail file, copy a system file, *delete a sounds file*, delete a memo file, delete a chart file, delete a spreadsheet file, *print a references file*, print a data file, print a graphics file, print a programming file, *save a text file to disk*, save a games file to disk, save a paint file to disk, save a statistics file to disk]

Schema sets (agent = designer)
SET U = [*drawing file*, news file, mail file, system file]
SET V = [*sounds file*]
SET W = [memo file, chart file, spreadsheet file]
SET X = [*references file*]
SET Y = [data file, graphics file, programming file]
SET Z = [*text file* games file paint file statistics file]

Rule schemata (agent = designer)
to copy SET U -> Click on SET U icon + click on OPERATION ICON 8
to delete SET V -> Click on SET V icon + click on OPERATION ICON 7
to delete SET W-> Click on SET W icon + click on OPERATION ICON 5
to print SET X -> Click on SET X icon + click on OPERATION ICON 2
to print SET Y -> Click on SET Y icon + click on OPERATION ICON 3
to save SET Z to disk -> Click on SET Z icon + click on OPERATION ICON 1

Schema sets (anticipated) (agent = user)
SET P = [*drawing file*, new file, mail file, system file]
SET Q = [*sounds file*, memo file, chart file, spreadsheet file]
SET R = [*references file*, data file, graphics file, programming file]
SET S = [*text file*, games file, paint file, statistics file]

Rule schemata (anticipated) (agent = user)
to copy SET P -> Click on SET P icon + click on OPERATION ICON 4
to delete SET Q -> Click on SET Q icon + click on OPERATION ICON 5
to print SET R -> Click on SET R icon + click on OPERATION ICON 2
to save SET S to disk -> Click on SET S icon + click on OPERATION ICON 1

Set compatibility
User's SET P matches designer's SET U (set compatible)
User's SET Q divided into designer's SET V and SET W (set incompatible)
User's SET R divided into designer's SET X and SET Y (set incompatible)
User's SET S matches designers SET Z (set compatible)

Rule compatibility
Rules match for: delete a memo file, delete a chart file, delete a spreadsheet file, *print a references file*, *save a text file to disk*, save a games file to disk, save a paint file to disk, save a statistics file to disk (Rule compatible)

Rule don't match for:*copy a drawing file*, copy a news file, copy a mail file, copy a system file, *delete a sounds file*, print a data file, print a graphics file, print a programming file (Rule incompatible)

Table 9.5. Formalisation of task-action rules associated with an experimental interface. Operation icons 1 to 8 are illustrated in figure 9.5 (the whole interface is illustrated in figure 9.6).

Note that users' schema sets have been predicted in the context of the tasks set. So, for example, drawing file, news file, mail file, and system file have been put in the same set (here labelled SET P) on the basis that subjects were asked to copy each of this type of file. Thus, if asked for a common semantic feature for the group, "files to be copied" might be a reasonable suggestion. Outside the context of the task set, it seems likely that users might put all objects into a single set, however the task set defines the context of each object's use in this experimental situation.

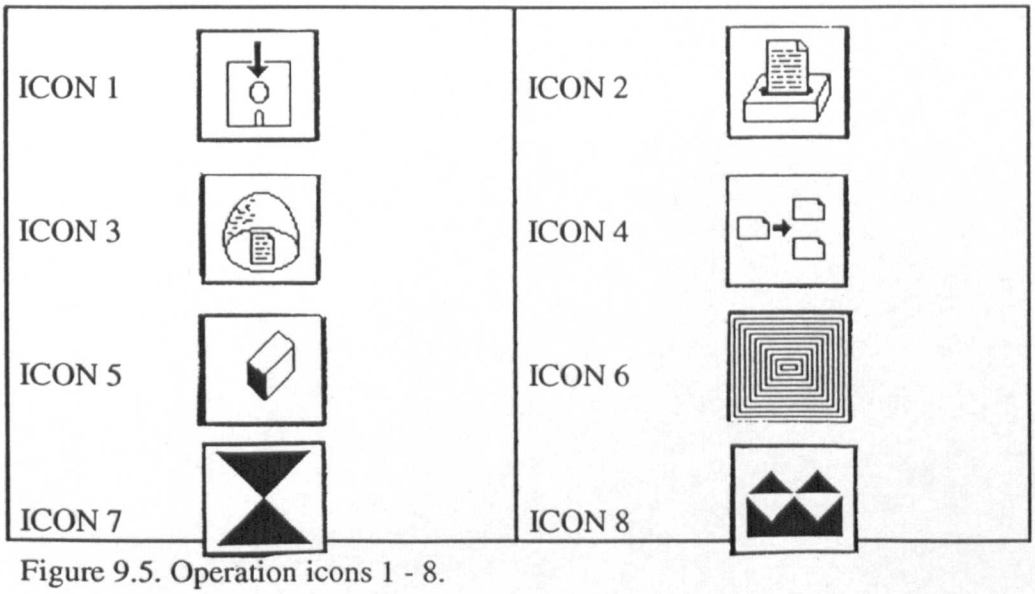


Figure 9.5. Operation icons 1 - 8.

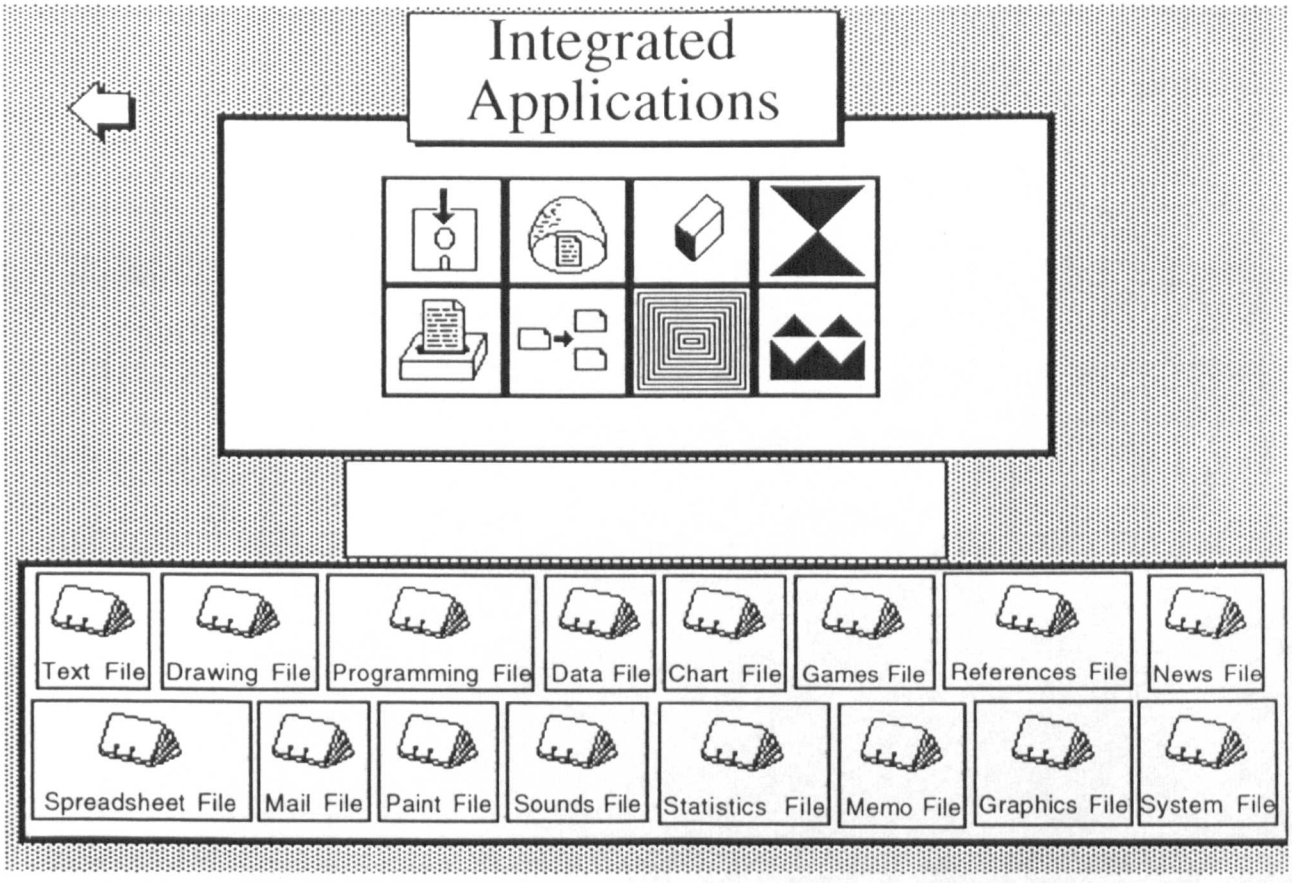


Figure 9.6. An experimental interface.

Experimental tasks by task type for the interface described were as follows (table 9.6).

Type A	Type B	Type C	Type D
save a text file	copy a drawing file	print a references file	delete a sounds file

Table 9.6. Experimental tasks by task type.

9.1.2 Subjects

Seventeen Glasgow University students participated in the study, nine males and eight females. All had prior experience of graphical interfaces and the Macintosh operating environment, but none were familiar with the specially designed

experimental interface. The icons used here are not used on the Macintosh anyway.

9.1.3 Experimental design

Comparisons were made within two groups of subjects. The first group (n = 8) were set the experimental tasks once each, in order that the predictions about guessability for users with no prior experience could be tested. The order in which these were set was controlled between subjects via a balanced Latin square design. The second group (n = 9) were set tasks in order to check the rest of the experimental hypotheses. Allocation to group was balanced for gender (one group contained four males and four females, and the other five males and four females). Subjects in the second group completed thirteen blocks of fillers, each followed by a block of experimental tasks. The first block of fillers contained twelve tasks (all the fillers once each), and subsequent blocks four (one filler per operation type). Performance on the second group's first experimental trial was to be used as a measure of the guessability of invoking a new command for those with previous experience of the interface — thus the comparatively substantial number of fillers in the first block. The order in which the fillers were set was randomised by "pulling numbers out of a hat". Experimental task order was balanced both between and within subjects using a balanced Latin square design. So, if one subject were to do a type A task first in the first experimental trial, another would do, say, a type C task first. Also, if a subjects first task in trial 1 were, for example, of type A then the first task in a later trial might be of, say, type D.

Performance was measured by number of errors and time on task. An error was said to have occurred when a subject clicked the mouse on an inappropriate operation icon — this would prompt an error message to appear on the display — whilst time on task was taken from when a subject received an instruction to when the message confirming successful task completion appeared.

9.1.4 Procedure

After reading an overview of the experiment (appendix 9.1) and an introduction to the experimental interface (appendix 9.2), and responding to a personal details interview, subjects were given a practice session. This involved clicking on icons

in a specially prepared practice interface, so that that they learned the order in which icons were clicked — object first, then operation — and so that they got used to the type of feedback that the interface gave. The experimenter "talked the subject through" the practice tasks. The practice interface did not contain any of the icons in the experimental interface. When subjects said that they were comfortable with the mechanics of the command activation, the experimental session was started.

The experimenter gave the subjects instructions in the form of (for example), "Please could you save a text file to disk". As soon as one task was complete, the next was set. Time on task and errors were recorded throughout the session. Subjects were asked to keep trying until they had completed each task. Throughout the session the experimenter followed a protocol (appendix 9.2) to ensure regularity of interaction with subjects. The experimental session took about an hour for those in the long condition, but only a couple of minutes for those in the short.

9.2 RESULTS AND DISCUSSION

Plots of performance by trial, in terms of time on task and errors, are illustrated in figures 9.7 and 9.8 respectively. The first point in these plots is taken from performance in the short experimental condition and is thus representative of performance for those with no previous experience of the interface. The other points are taken from the second point measured in the long condition onwards. This was done for purposes of comparison with the plots from the experiment in chapter 7. Note, however, that here there is a comparatively large difference between trials one and two in terms of the number of fillers that users have attempted by this stage.

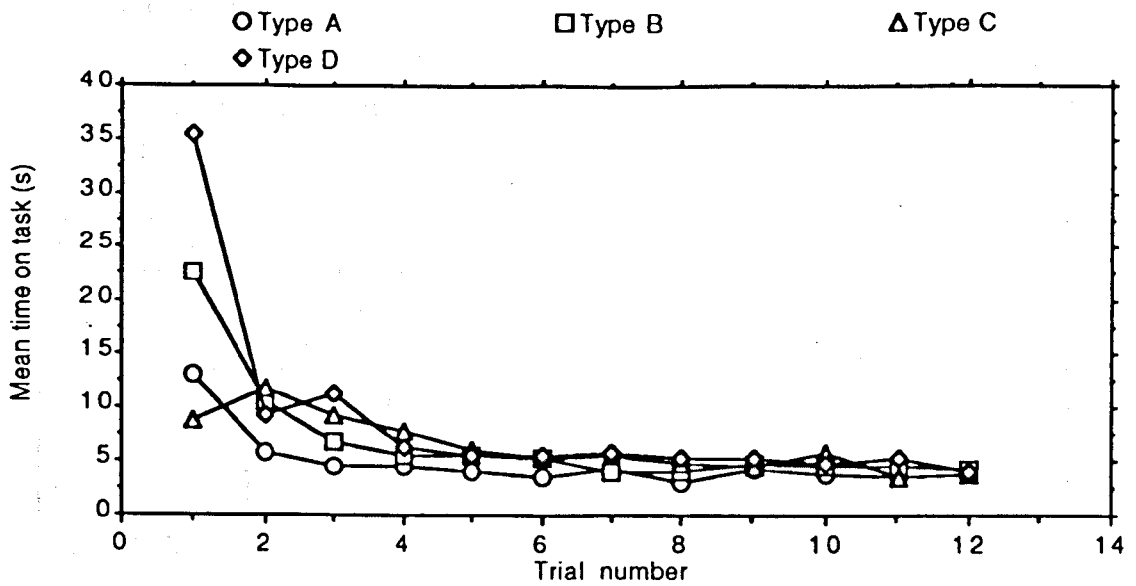


Figure 9.7. Mean time on task (seconds) per trial for each task type.

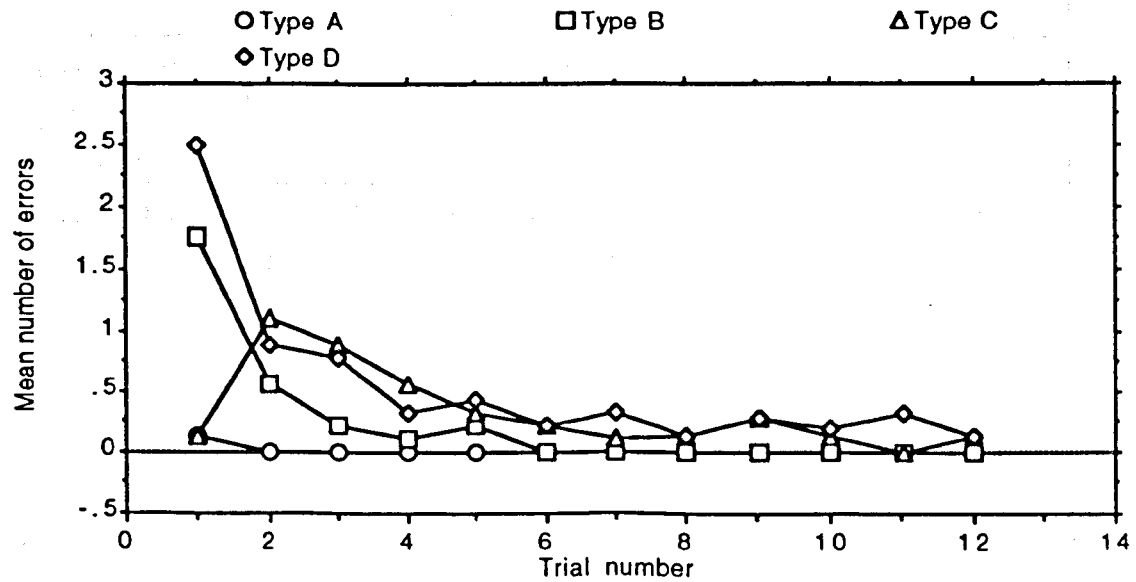


Figure 9.8. Mean number of errors per trial for each task type.

9.2.1 Guessability

The number of errors made and the time taken to complete tasks by the first group of subjects (n = 8) was taken to be indicative of guessability for users with no

previous experience of the interface (see table 9.7). Data from the second group's (n = 9) first experimental trial was taken as a measure of guessability for those with previous experience on similar tasks (see table 9.8). The performance of each group is illustrated in terms of time on task and errors respectively, in figures 9.9 and 9.10.

MEASURE	Type A	Type B	Type C	Type D
Errors	0.13 (sd = 0.35)	1.75 (sd = 1.28)	0.13 (sd = 0.35)	2.50 (sd = 1.51)
Time on task (seconds)	12.9 (sd = 6.5)	22.7 (sd = 9.1)	8.7 (sd = 3.2)	35.5 (sd = 13.1)

Table 9.7. Mean guessability values (with standard deviations) by task type, for subjects with no previous experience of the interface (n = 8).

MEASURE	Type A	Type B	Type C	Type D
Errors	0.11 (sd = 0.33)	0.11 (sd = 0.33)	1.89 (sd = 1.27)	3.11 (sd = 2.09)
Time on task (seconds)	6.3 (sd = 1.0)	7.4 (sd = 1.7)	20.1 (sd = 13.1)	32.4 (sd = 31.9)

Table 9.8. Mean guessability values (with standard deviations) by task type, for subjects who had previous experience with the interface (n = 9).

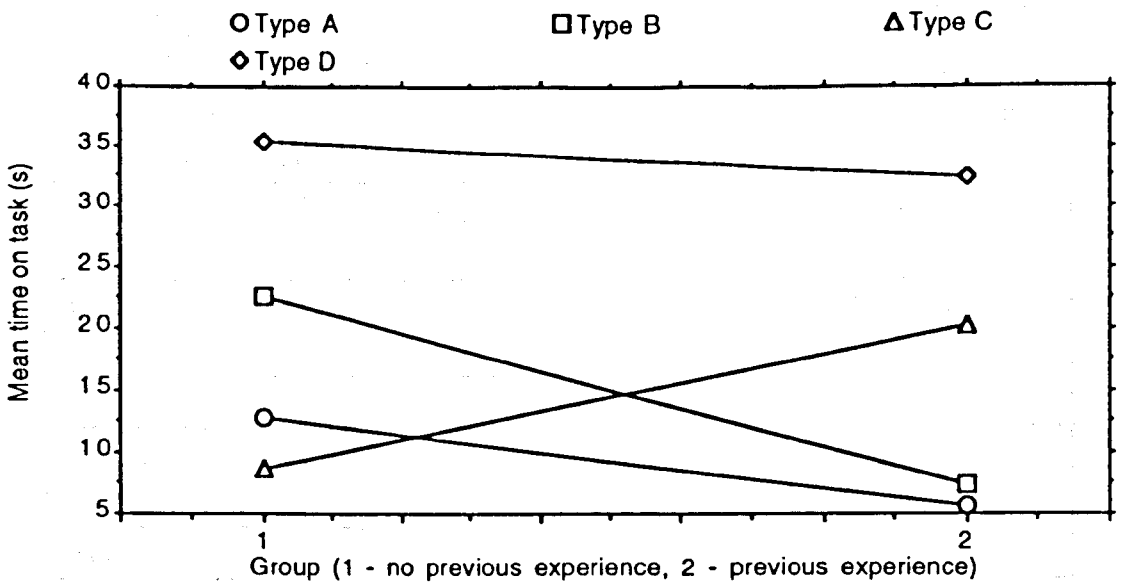


Figure 9.9. Mean time on experimental task (seconds) for subjects without (n = 8) and subjects with (n = 9) previous interface experience on filler tasks.

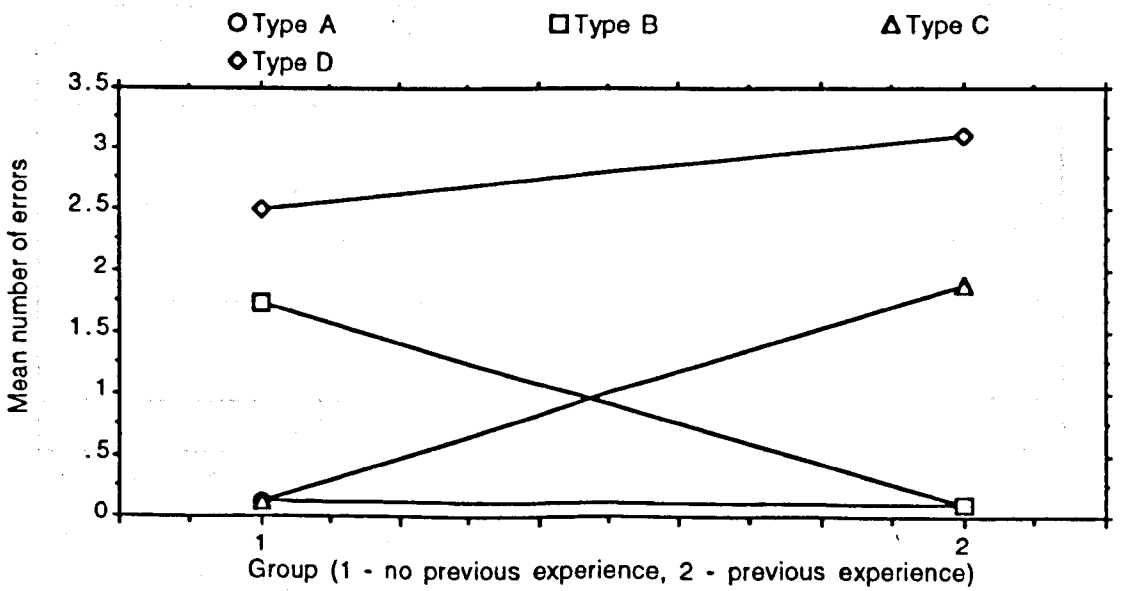


Figure 9.10. Mean number of errors on experimental task (seconds) for subjects without (n = 8) and subjects with (n = 9) previous interface experience on filler tasks.

The predictions were checked via Wilcoxon signed rank (WSR) tests, to check for statistically significant differences. Results of the tests are summarised in tables 9.9 and 9.10. Remember that A > B, for example, represents the prediction that users will perform better on task type A than on task type B (take less time and make fewer errors).

Comparison	Prediction	Errors	Time on task (s)
A vs C	A = C	NS, T = 1.5, n = 2	NS, T = 12
B vs D	B = D	NS, T = 9, n = 7	NS, T = 10
A vs B	A > B	$p < 0.02$, T = 0, n = 7	$p < 0.10$*, T = 5
C vs D	C > D	$p < 0.01$, T = 0	$p < 0.01$, T = 0

Table 9.9. Comparison of guessability values between task types for users without previous experience of the interface (n = 8). Where sample sizes have been reduced due to extraction of tied values, this has been indicated. *marks marginal significance.

Comparison	Prediction	Errors	Time on task (s)
A vs C	A > C	$p < 0.02$, T = 0 (n = 7)	$p < 0.01$, T = 0
B vs D	B > D	$p < 0.01$, T = 0, n = 8	$p < 0.01$, T = 0
A vs B	A = B	NS, T = 1.5, n = 2	NS, T = 9
C vs D	C > D	*†NS, T = 12*	*†NS, T = 10*

Table 9.10. Comparison of guessability values between task types for users without previous experience of similar tasks (n = 9). Where sample sizes have been reduced due to extraction of tied values, this has been indicated. *marks marginal significance.† Marks failure of prediction.

Results were largely as predicted, the exceptions being that the difference between time on task for types A and B for inexperienced users was only marginally significant, and the lack of a statistically significant difference between performance on task types C and D for those with previous experience. The sample sizes used here were fairly small in comparison with those used in the other studies reported so far — this reduces the chance of statistically significant

differences being found. Perhaps, then, results from WSR tests should not be considered in isolation from mean performance values. For example, even though the difference between performance on task types C and D for those with previous interface experience is not statistically significant, the results listed in table 9.8 indicate that, in terms of means, there was a fairly large discrepancy. Type D tasks took a mean of 65% longer than type C, and had 61% more errors associated with them.

As with the two studies using textually labelled menu based interfaces, the results support the idea that for those with no previous experience of an interface rule compatibility affects guessability more strongly, but that for those with experience of similar tasks set compatibility may be the more important factor.

9.2.2 Learnability

A value for learnability was derived from how many trials it took for subjects to establish error free performance. The first trial was not considered in this part of the analysis as it had been used for guessability data. As in the study reported in chapter 7, subjects were said to have reached error free performance from the trial subsequent to their last error on any particular type of task. So, if a subject were to make an error on a task in, say, trial number 6, but made no more during the remainder of the experimental session, then they would be deemed to have reached error free performance on that task by trial 7. After subtracting the first (guessability) trial, this would leave a learnability score of 6 trials for the subject on that task type.

In terms of time on task, it was decided, for the sake of meaningful comparison, that being able to complete a task in less than six seconds would be regarded as being a competent level of performance. From glancing at the data, it was apparent that all subjects were able to attain this level of performance on most tasks by the end. However, the number of trials before this was reached appeared to vary between tasks. The trial at which competence was said to have been reached, was the first of two consecutive ones in which performance reached this level. This guarded against "lucky strikes" early on, where performance might reach this level on a one-off basis. By not insisting that performance remain at this level on a task for every subsequent trial a safeguard was also established against the possibility

of distortions due to, say, a temporary lapse in concentration. Again, the first (guessability) trial was excluded from this analysis.

If a subject failed to reach the criterion value on a task, a default value of thirteen trials was set. This assumes that had the session continued they would have fulfilled the criterion by the next trial. Again, whilst there is no reason to expect this assumption to be valid, setting such a default value gives a lower bound on learnability values.

Two of the subjects did not complete the full thirteen trials during the experimental session. In terms of, for example, errors, had their last few trials been error free, a learnability value was set from the trial after which they made their last error. If, however, they were still making errors on a particular type of task by the end of the session, a default value was set based on the number of trials they had completed. Again this was based on the assumption that the relevant criterion would have been met on the next trials had the session continued. A similar approach was taken with respect to time on task. Again, these default values act as a lower bound. Despite the distortions to the means that may be caused by setting these values, they can still be useful in non-parametric tests for significant differences.

Results are summarised in table 9.11.

Measure	Type A	Type B	Type C	Type D
Trials to time < 6 s	2.75 (sd = 1.04)	4.63 (sd = 1.92)	6.00 (sd = 3.11)	5.13 (sd = 3.44)
Trials to error free performance	1.00 (sd = 0.00)	1.89 (sd = 1.36)	5.44 (sd = 3.71)	6.44 (sd = 5.32)

Table 9.11. Mean learnability values (units: trials).

As usual WSR tests were used to test the experimental predictions. Outcomes are summarised in table 9.12.

Comparison	Prediction	Time on task	Errors
A vs C	A > C	p < 0.02, T = 0 (n = 7)	p < 0.02, T = 0
B vs D	B > D	*†NS, T = 7.5 (n = 6)*	p < 0.05, T = 2 (n = 7)
A vs B	A > B	p < 0.10, T = 1.5 (n = 6)	*†NS, T = 0 (n = 3)*
C vs D	C > D	*†NS, T = 4 (n=6)*	*†NS, T = 13 (n = 8)*

Table 9.12. Comparison of learnability values between tasks types using WSR tests (n = 9, unless stated otherwise. † Marks failure of prediction.

Again, the small sample size suggests that it might be unwise to look at outcomes of WSR tests in isolation from mean values. Nevertheless, it appears that whilst there is a clear effect for set compatibility, the results indicate that the issue is less clear cut for rule compatibility. Mean values indicate that where there is set compatibility, having rule compatibility also will further enhance learnability (A vs B). However, where there is no set compatibility, effects for rule compatibility seem to disappear.

So why should this be the case? A possible explanation is that, whilst both set and rule compatibility affect learnability, it is set compatibility which has the bigger effect (the results of the study reported in chapter 7, as well as this study indicate this). It may be, then, that increases in time on task and errors due to lack of set compatibility have swamped effects due to rule compatibility. Thus, effects for rule compatibility which are apparent where there is set compatibility (A vs B) may not be apparent in its absence (C vs D).

The errors in the design of the experimental interface may also have been a factor here. As explained in the apparatus section, type D tasks were, perhaps, not as strongly rule incompatible as they might have been, due to the unsuitability of the symbols on the icons used for filler tasks. This might mean that it was comparatively easy to "un-learn" any initial misconceptions about the icon required. On closer inspection, however, this objection appears untenable. After all, there appears little doubt that type D tasks are still less rule compatible than the type C ones with which they are being compared, as the icons used for type C tasks were highly suitable. This is supported by the huge differences in

guessability for subjects with no previous experience with the experimental interface — type C tasks were more than four times as guessable in terms of time on task, and over nineteen times as guessable in terms of errors. In fact these subjects performed worse, on average, with type D tasks than with type B tasks, which were supposedly more strongly rule incompatible. Furthermore, this experimental prediction (i.e. that type C tasks are more learnable then type D), received little support from either of the other studies in which it was tested (see chapters 7 and 10).

9.2.3 EUP

Performance on the later trials in the experimental session is illustrated in figures 9.11 and 9.12.

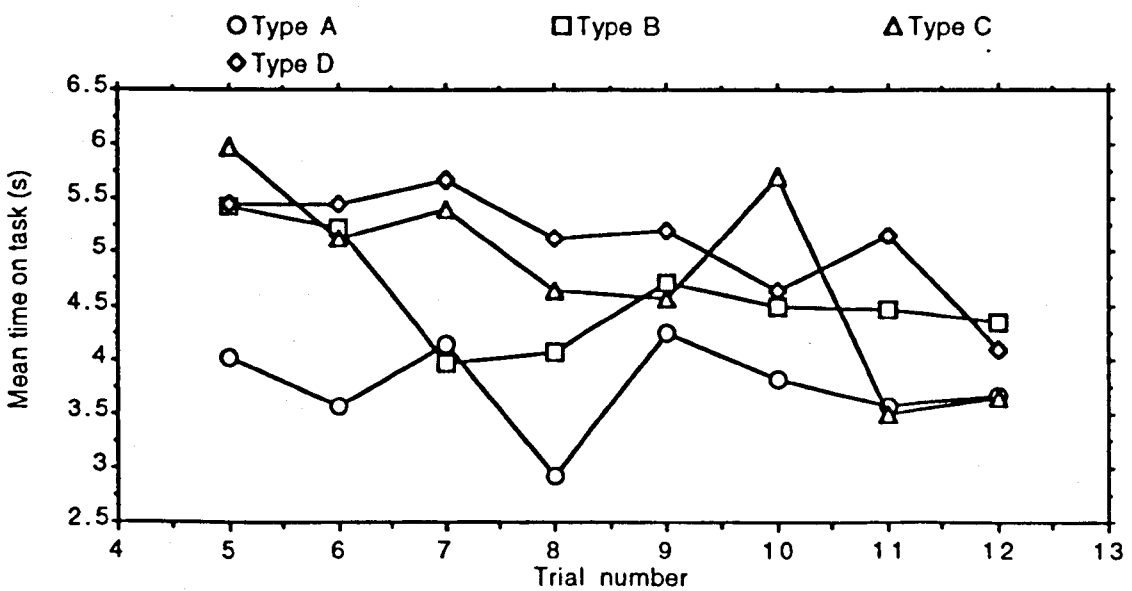


Figure 9.11. Mean time on task (seconds) over the last few trials for each type of task.

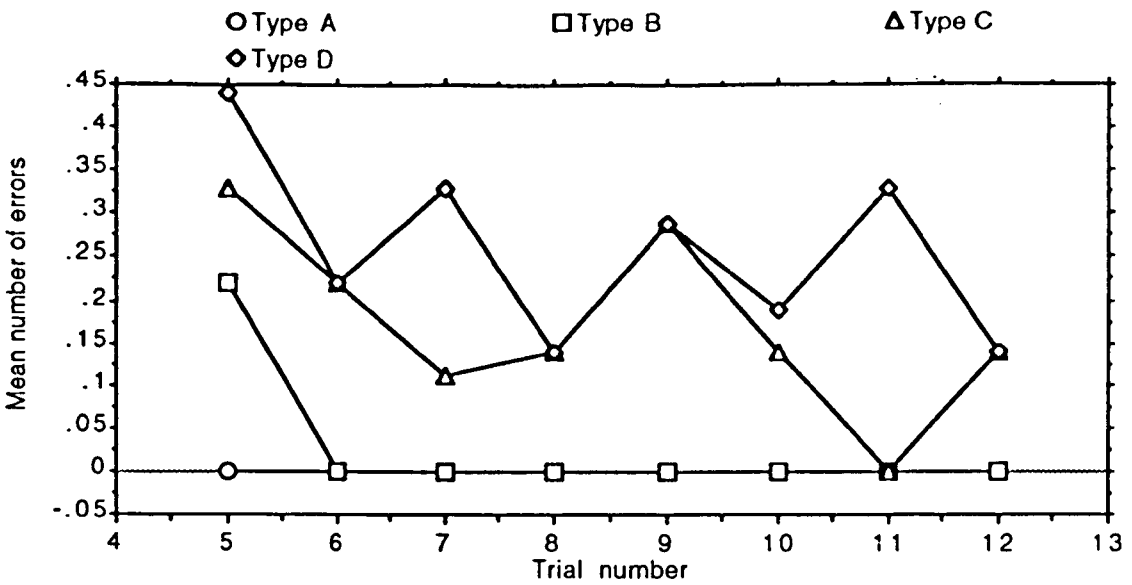


Figure 9.12. Mean number of errors over the last few trials for each type of task.

Performance over the last three trials was taken to represent EUP. Mean EUP values are listed in table 9.13. Because the error rate was comparatively low, the figures quoted refer to the mean of subjects' total errors over the last three trials. The time on task figures, however, refer to the mean times per trial. Data from only seven subjects was considered in the EUP analysis, as two did not get as far as trials eleven twelve and thirteen. Visual scrutiny of the performance curves suggests that mean performance has steadied by this stage. However, as a check a linear regression equation was fitted to the last four points of a mean plot of cumulative number of tasks completed vs. cumulative time. Associated R-squared values were 1.000, 0.998, 0.999, and 0.997, for task types A, B, C, and D respectively.

MEASURE	Type A	Type B	Type C	Type D
Time on task	3.72 s (sd = 0.95 s)	3.97 s (sd = 1.77 s)	3.51 s (sd = 0.91 s)	4.92 s (sd = 4.34 s)
Errors	0.00 (sd = 0.00)	0.00 (sd = 0.00)	0.29 (sd = 0.76)	0.86 (sd = 0.1.46)

Table 9.13. Mean EUP values.

Because only four subjects made errors (three on type D tasks, and one on type C), it would not have been possible to find statistically significant differences from

a WSR test in terms of this measure. Also, WSR tests gave no statistically significant differences for task type in terms of time on task.

Whilst, again, analysis for statistical significance may not be the most meaningful way of checking the predictions, these results still appear inconclusive, although the differences in the error data are in the predicted direction. As expected the few errors which did occur, were all made when performing set incompatible tasks. The mean times on task, however, appear to show no pattern at all — mean times to complete types A, B, and C all being within half a second of each other, whilst mean time to complete type D was distorted by the performance of one subject who had an EUP of almost 15 seconds for this type of task (the subject made a total of four errors on the type D task during the last three trials).

A possible reason for the lack of inter-task differences may have been experimental noise, which could, given the comparatively short times, have smothered any possible effects. There were two possible sources of noise. Firstly, the time subjects had to spend searching for their target icons on the display. In studies 2 and 3 there were only seven menu headings to scan, all arranged along the top of the interface, whereas in this study subjects had to spot the object icon amongst 15 others. Furthermore, all object icons looked similar, differing only in their written labels. Another potential source of noise was a variable interface response time to user inputs. For a reason unknown to the investigators, feedback from the interface appeared to be different for different tasks. This caused problems as time on task was taken from the subject receiving an instruction to when the confirmation message appeared. In retrospect, it would have been preferable if the experimenter had stopped the watch when the subject clicked the operation icon. However, as there were four different configurations of interface to remember, the experimenter was not always sure if the subject had clicked the correct icon until a message appeared.

9.2.4 System potential

There should have been no noticeable differences between the system potentials of any of the tasks set. Even if there had been these would have been evened out over the four interfaces as task types were balanced for operation (i.e. if the type A experimental task involved deleting in one of the experimental interfaces, it might involve, say, saving in another).

9.3 SUMMARY

The predictions about the effects of set and rule compatibility on guessability, learnability, and EUP, having previously been tested using menu based interfaces, have now been investigated in the context of an iconic interface. Here, then, consistency was connected with the usage and suitability of icons with pictorial labels, whereas as in the previous studies, consistency was connected with associations between textual command names and menu headings. Although the smaller sample sizes used here meant that it was not always possible to find statistically significant effects, mean values appeared to support nearly all of the predictions. The only clear exception being, that rule compatibility was not shown to aid learnability in situations where set compatibility was absent. Overall, however, results suggest that the effects for consistency found with the menu based interfaces may be generalised to an iconic one.

Chapter 10

APPLYING THE THEORY TO A COMMAND DRIVEN INTERFACE

So far, the predictions about the effects of set and rule compatibility on the various components of usability have been tested via experiments with two different types of operating environment — text menu based, and iconic. This chapter reports a study where a command driven interface was used.

Command driven interfaces differ from menu based and iconic interfaces, in that they are recall rather than recognition based. This means that a user is required to infer or remember the command name for a particular operation, rather than having to choose or recognise it from a set of options. A study by Mayes, Draper, McGregor, and Oatley (1988) looked at experienced users of the recognition based Apple Macintosh operating environment. They found that users could often recall rather little about how to perform what, for them, were fairly routine tasks, yet this appeared not to affect their performance. However, users of recall based interfaces must remember the names of commands and arguments, as well as, for example, the order demanded by the syntax. Much of the previous work on consistency in HCI was done solely on recall based (command line) user interfaces.

The study reported here investigated users' ability to infer and remember the command name for a particular operation. The aim was to be able to make a statement about the prospects for generalising the predictions tested in the preceding studies to include recall based environments.

Guessability (with no previous experience of the interface), learnability, EUP, and re-usability were investigated for four types of task (see table 10.1).

	RULE COMPATIBLE	**RULE INCOMPATIBLE**
SET COMPATIBLE	Type A	Type B
SET INCOMPATIBLE	Type C	Type D

Table 10.1. Set and rule compatibility of experimental tasks.

The experimental hypotheses, based on the predictions made in chapter 6 were as follows (table 10.2).

COMPONENT OF USABILITY	EFFECT OF SET COMPATIBILITY	EFFECT OF RULE COMPATIBILITY
Guessability	A = C, B = D	A > B, C > D
Learnability	A > C, B > D	A > B, C > D
EUP	A > C, B > D	A = B, C = D
Re-usability	A > C, B > D	A > B, C > D

Table 10.2. Experimental hypotheses.

The approach taken was to ask subjects to invoke commands to perform editing operations, where each command string contained a command name and an argument. The command name specified the operation to be performed, whilst the argument specified the object operated on. So, for example, a user might have to type "DELETE TEXT" and then press return. Overall, subjects were asked to perform four different types of operation, each on two different objects.

Set compatibility was manipulated according to whether or not the same command name was used for a particular operation irrespective of the object to be operated on, or whether the command name was object dependent. For example, if the commands for erasing some text and erasing a table were "DELETE TEXT" and "DELETE TABLE", respectively, these would be set compatible. However, if they were "DELETE TEXT", and "REMOVE TABLE", they would be set incompatible. Rule compatibility, meanwhile, concerned the a priori appropriateness of a command name. So, if the command for erasing text were "DELETE TEXT" this would be comparatively rule compatible, but if the command name were a nonsense syllable, giving a command string such as, say, "BLEE TEXT" it would not. Consistency here, then, is essentially about the meaning similarities in ordinary English between alternative command names, and between command name and the wording of the task instruction.

10.1 THE STUDY

10.1.1 Apparatus

A simulation of a command driven interface was created using the word finder facility of Microsoft Word 4 for the Apple Macintosh. This facility searches through a document in order to find words — the words being specified by the user typing them into an entry field. If the words are found, they are highlighted, and the area of the document around them displayed. In this study, users were presented with the opened entry field, and told that this was where they were to type in commands. If they typed an appropriate command string (including command name and argument name), the document would be scrolled to where the name appeared in the text, under which would be a message telling them the operation which they had performed. If, however, they typed an inappropriate string, a message "end of the document reached" would appear. Subjects were told that this meant the machine had searched through a "library" of commands, but couldn't find the one they typed, and was thus unable to invoke it.

In total four types of operation were simulated, each on two different object types: deleting (text and tables), formatting (pages and graphics), inserting (words and letters), and saving (files and lines). Performing operations on the first objects in the above pairs constituted the experimental tasks, whilst filler tasks came from performing operations on the second objects. Four configurations of interface were created for the experiment. An experimental task of each type (A, B, C, and D) could be performed using each interface (comparisons were within subjects). The assignment of experimental tasks to task type was balanced across interfaces, via a Latin square. Table 10.3 lists formalisations of the task-action rules associated with one of the experimental interfaces.

Taskset
TASKSET = [delete text, delete table, format page, format graphic, insert word, insert letter, save file, save line]

Schema sets (agent = designer)
SET U = [TEXT, TABLE]
SET V = [PAGE]
SET W = [GRAPHIC]
SET X = [WORD]
SET Y = [LETTER]
SET Z = [FILE, LINE]

Rule schemata (agent = designer)
to delete U -> type "DELETE" + type U + press return
to format V -> type "FORMAT" + type V + press return
to format W -> type "STYLE" + type W + press return
to insert X -> type "BAZ" + type X + press return
to insert Y -> type "FOO" + type Y + press return
to save Z -> type "BLEE" + type Z + press return

Schema sets (anticipated) (agent = user)
SET P = [TEXT, TABLE]
SET Q = [PAGE, GRAPHIC]
SET R = [WORD, LETTER]
SET S = [FILE, LINE]

Rule schemata (anticipated) (agent = user)
to delete P -> type "DELETE" + type P + press return
to format Q -> type "FORMAT" + type Q + press return
to insert R -> type "INSERT" + type R + press return
to save S -> type "SAVE" + type S + press return

Set compatibility
User's SET P matches designer's SET U (set compatible)
User's SET Q divided into designer's SET V and SET W (set incompatible)
User's SET R divided into designer's SET X and SET Y (set incompatible)
User's SET S matches designer's SET Z (set compatible)

Rule compatibility
Rules match for: delete table, delete text, format page (rule compatible)

Rules do not match for: format graphic, insert word, insert letter, save file, save line (rule incompatible)

Table 10.3. Formalisation of task-action rules associated with an experimental configuration.

The "nonsense syllables" used for operation argument names in the rule incompatible command strings were based on command names suggested by Lewis, Hair, and Schoenburg (1989). Note, that users anticipated schema sets have been devised in the context of the tasks set. So, for example, users might think of the objects in SET P as "items to be deleted", or those in SET Q as "items to be formatted". Outside the context of the experiment, there is no reason to

suspect that users would group items in this way. Indeed it seems likely that they might regard them all as being members of a single schema set.

For this configuration the experimental tasks, by task type, were as listed in table 10.4.

EXPERIMENTAL TASK	TYPE	(FILLER)
delete text	A	(delete table)
format page	C	(format graphic)
insert word	D	(insert letter)
save file	B	(save line)

Table 10.4. Experimental tasks by task type.

The interfaces also included help facilities. If users wanted to find the command string for a particular task, they first had to type "HELP". A message would then appear asking which operation they wanted help with — they were asked to type either deleting, formatting, inserting, or saving. (The reason for using the same English word in both help and the experimenter's instructions to subjects, as was used for the command name, is discussed later, in the "Procedure" section). Having typed this in they were then asked to specify the object to be operated on. So, for example, if a subject wanted help with deleting they would then have to specify whether they wanted to delete text or a table. The help facility was deliberately cumbersome to use — it would have been quicker for subjects had it been designed so that, say, a single string such as "HELP/DELETING/TEXT" could produce the relevant information. The intention was to prevent subjects adopting a strategy reliant on using help throughout the experimental session, which may have meant some would not bother to learn command strings. This would have undermined the aim of the study, as performance would no longer rely on recall.

10.1.2 Subjects

Eight female Glasgow University students participated in the study. All had used a computer before, and all but one had word processing experience. Five had used the Apple Macintosh before, and three of these had used the Macintosh version of Microsoft Word. Of the three subjects who had not used the Apple Macintosh

before, the two with word processing experience had used a version of Word with IBM compatible machines. However, as the experimental interface was designed to simulate a command string driven operating environment, it was not expected that previous use of Word in a normal context would affect performance here.

10.1.3 Experimental design

Two subjects were allocated to use each interface configuration. This was done randomly by "drawing numbers from a hat". Each subject attended two experimental sessions. In the first they completed twelve experimental trials, each followed by a block of filler tasks. Each trial consisted of performing all four experimental tasks, and each filler block contained all the filler tasks. In the second subjects performed a further experimental trial.

The order in which the experimental tasks were attempted was balanced both across and within subjects, using a Latin square design. As with previous studies, this meant that if one user were to attempt a type A task first on her first trial, then another might attempt, say, a type C task. Also, if a subject had attempted a type D task first in her first trial, she might attempt, say, a type C task first in her next trial.

Performance was quantified both in terms of time and errors. Because consistency issues were connected with the command names used, errors were classified as typing in incorrect command names. Typing mistakes were not counted as errors, nor were mistakes in the order in which command name and argument were typed in, as these were irrelevant in the context of the experimental design — these sorts of mistake, however, might be a source of noise with respect to the time on task data.

10.1.4 Procedure

After reading an overview of the experiment (see appendix 10.1), subjects were given a verbal introduction to the experimental interface. They were told that in order to perform a task they would have to type in a two part command string, specifying a command name and an argument, with the command name being typed first. Subjects then used a separate practice interface, in order to familiarise

themselves with the general principles of how the interface worked. They were instructed to type strings into the entry field, for example XXX YYYYY, and press return. A message would then appear, either telling the user that they had performed some sort of operation on an object, or "end of document reached", depending on the string typed. Subjects were also shown how to use the help facility — again using the practice interface. Subjects were told that although meaningless strings of letters were used in the practice interface, this would not necessarily be the case in the experimental interface.

The first experimental session, which lasted about an hour and a half, was then started. Subjects had no practice with the experimental interface before the first trial was set. Instructions were given in the form of: "Please could you invoke the command for deleting some text". Note that the phrasing of the instructions was designed to make the appropriate command names as rule compatible as possible. If, for example, the experimenter had asked the subject to "erase some text", then, in the light of the way the instruction was phrased, the command name "DELETE" may not, in fact map onto the first a priori expectations of users. This is because, as previous work (Furnas, Landauer, Gomez, and Dumais 1983) has shown, there is rarely a command name for a particular operation which is agreed on by a high percentage of a user population. This implies that any given name is unlikely to be rule compatible for a population as a whole. Had users been asked to "erase" objects, then "DELETE" would still be more compatible than "BLEE", but it might easily be, say, the third or fourth guess rather than the first. So, there might still have been an effect even if instructions had been given in a different form. However, there seemed a danger that these might not show up in the experimental context unless rule compatibility were maximised. For example, on the first trial, if users didn't get the operation name right after a couple of guesses, they might simply have reverted to help, in which case the data would not show a third or fourth choice name to be any more guessable than a nonsense-syllable. (Intuitively it seems more likely that effects for weaker rule compatibilities might be more likely to show up as effects on learnability rather than on guessability. So whilst "DELETE" may not be very guessable for the instruction "erase", it should be easier to recall than a nonsense-syllable once used).

As soon as one trial and block of fillers was complete, another was set without delay, until twelve had been completed. Approximately twenty four hours later subjects returned to complete another trial. The experimenter followed a written protocol throughout the session (appendix 10.2).

During the experimental sessions the experimenter noted any errors made by subjects, and any help consultations they made. A video record was kept of each experimental session, which was later replayed for time on task analysis. Time from the end of the experimenter's instruction to the typing of the first letter of the correct operation argument name was recorded. Time to task completion was not used as noise due to, say, the different length of command names, or spelling errors (neither of which are consistency issues), might hide effects (noise appeared to be a problem at EUP in the study with the iconic interface reported in the previous chapter). During analysis investigator judgement was used to determine whether any apparent errors were typing errors, or due to genuine confusion as to what the argument names were. In practice, subjects nearly always noticed typing errors and corrected them. Measurements of time to starting typing, were repeated on the trials to be used as measures of EUP, until values obtained were within + or - 0.05 seconds of each other. This was nearly always achieved with just one repetition. Where differences were found, the mean of the two measures was used. It was important that these times were measured accurately as subjects were often taking less than a second to get started with typing the command names, therefore any noise from experimenter response times might smother consistency effects. Tape analysis took about an hour and a half per subject.

10.2 RESULTS AND DISCUSSION

Mean performance by task type by trial is illustrated in terms of time on task, errors, in figures 10.1, 10.2 respectively. The number of subjects consulting help is illustrated in figure 10.3.

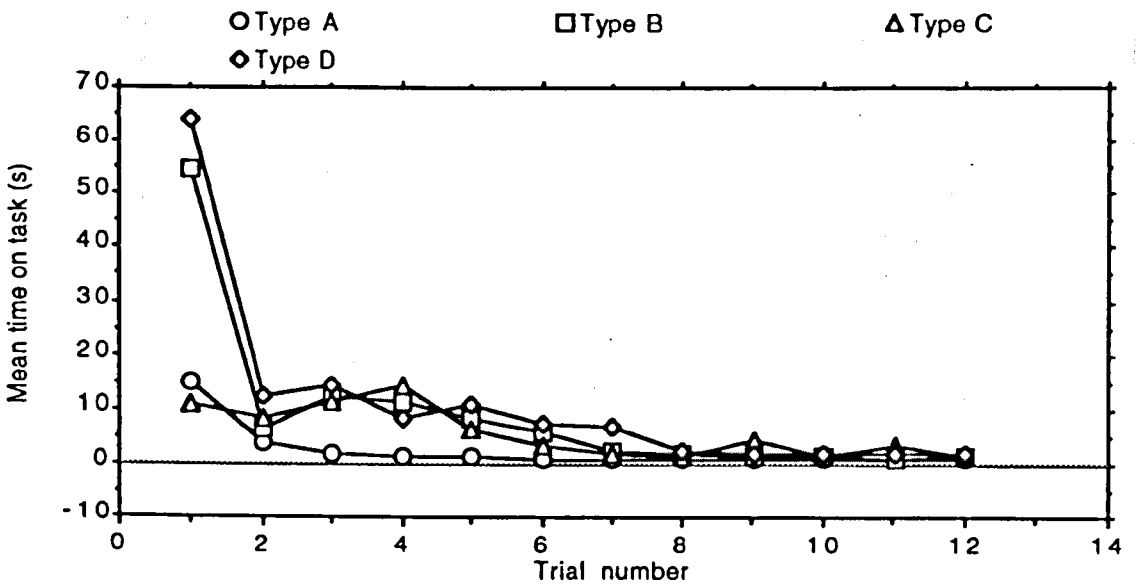


Figure 10.1. Mean time on task (seconds) per trial for each task type.

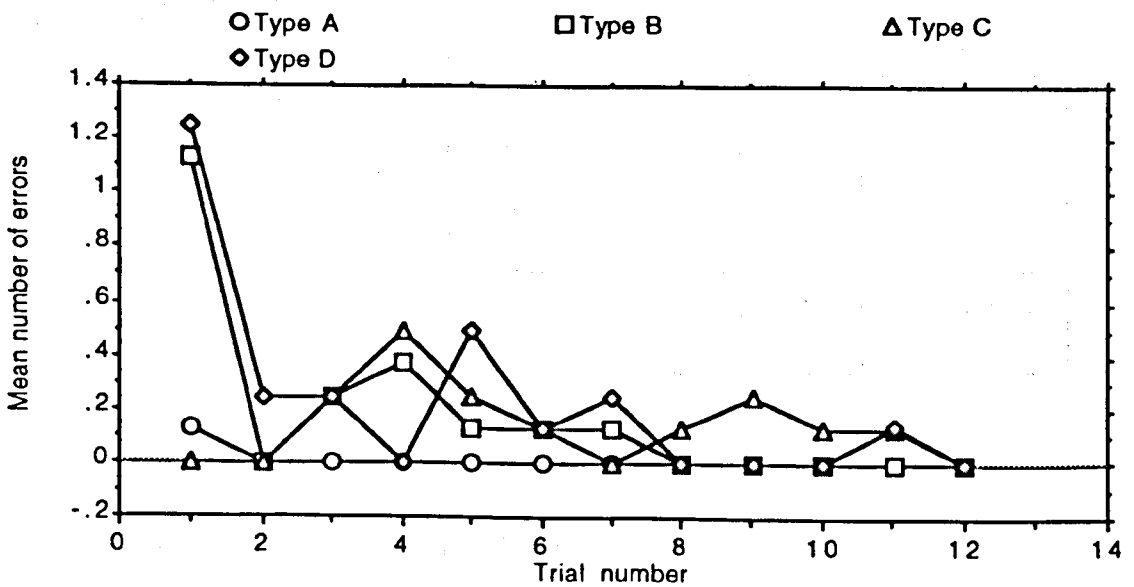


Figure 10.2. Mean number of errors per trial for each task type.

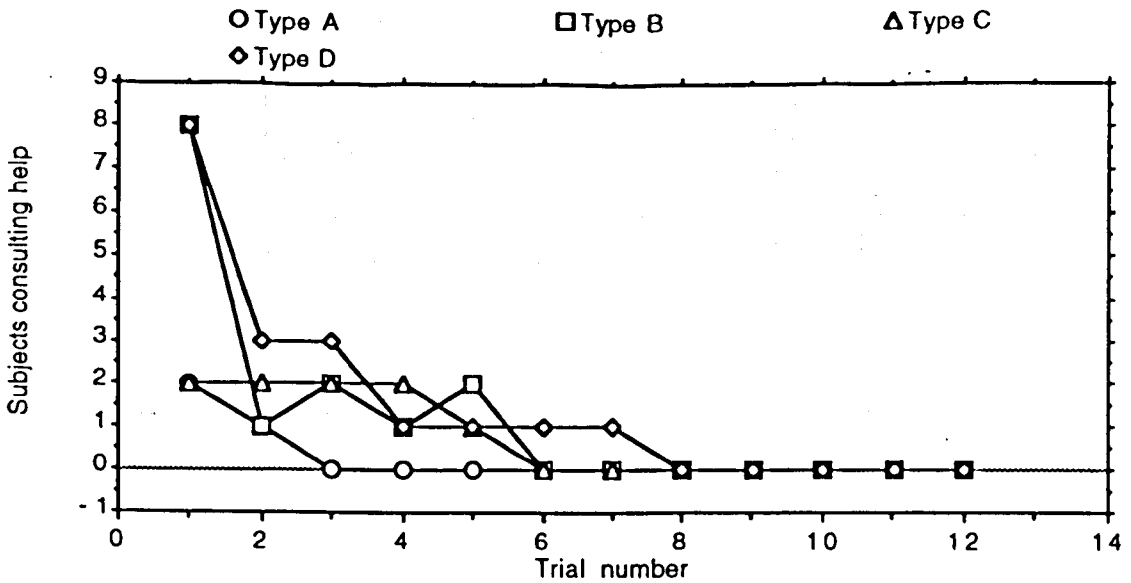


Figure 10.3. Number of subjects consulting help per trial for each task type.

10.2.1 Guessability

Performance on the first trial of the first experimental session was taken as being representative of guessability. Mean guessability values by task type are listed in table 10.5. The help consultation value is given in terms of how many of the eight subjects used the facility on the first trial.

MEASURE	Type A	Type B	Type C	Type D
Time on task	15.0 s (sd = 19.2 s)	54.4 s (sd = 18.8 s)	10.6 s (sd = 12.2 s)	63.9 s (sd = 28.1 s)
Errors	0.13 (sd = 0.35)	1.13 (sd = 0.99)	0.00 (sd = 0.00)	1.25 (sd = 0.71)
Help Consultations	2 subjects (n = 8)	8 subjects (n = 8)	2 subjects (n = 8)	8 subjects (n = 8)

Table 10.5. Mean guessability values.

Wilcoxon signed rank (WSR) tests were used to check the predictions in terms of time on task and errors. Table 10.6 illustrates that the results were as predicted, with a strong and statistically significant effect being found for rule compatibility.

COMPARISON	PREDICTION	TIME ON TASK	ERRORS
A vs C	A = C	NS, T = 9	NS, T = 1, (n = 1)
B vs D	B = D	NS, T = 15	NS, T = 2.5, (n = 3)
A vs B	A > B	$p < 0.01$, T = 0	*$p < 0.10$, T = 3, (n = 7)
C vs D	C > D	$p < 0.02$, T = 1	$p < 0.02$, T = 0 (n = 7)

Table 10.6. Comparison of guessability between task types (n = 8, unless otherwise stated). * indicates marginally significant.

10.2.2 Learnability

As with the studies reported in preceding chapters, the trial after the last in which an error occurred was taken as representative of learnability. The criterion set for learnability in terms of time was that subjects should type the first letter of the appropriate command name within two seconds of receiving an instruction for two consecutive trials. In terms of help consultations, the trial after which a subject made her final consultation was used. As usual the first (guessability) trial was not considered in the learnability analysis. If a subject did not reach a set criterion on any of the tasks, a default value of twelve trials was set. Three subjects failed to meet the learnability criterion, in terms of time, for one type of task each (two on task type D and one on type C). Mean learnability values are summarised in table 10.7.

MEASURE	Type A	Type B	Type C	Type D
Time on task	2.00 (sd = 1.07)	5.75 (sd = 2.38)	6.25 (sd = 3.15)	7.00 (sd = 4.18)
Errors	1.00 (sd = 0.00)	3.13 (sd = 2.48)	4.63 (sd = 3.96)	4.38 (sd = 3.42)
Help consultations	1.13 (sd = 0.35)	2.13 (sd = 2.10)	2.13 (sd = 1.64)	2.38 (sd = 2.07)

Table 10.7. Mean learnability values (units: trials).

The results of WSR tests for statistical significance are given in table 10.8.

Comparison	Prediction	Time on task	Errors	Help consultations
A vs C	A > C	p < 0.02, T = 0	p < 0.10, T = 0 (n = 5)	*NS†, T = 0 (n = 3)*
B vs D	B > D	*NS†, T = 6 (n = 6)*	*NS†, T = 3.5 (n = 6)*	*NS†, T = 3.5 (n = 4)*
A vs B	A > B	p < 0.02	*NS†, T = 0 (n = 4)*	*NS†, T = 0 (n =2)*
C vs D	C > D	*NS†, T = 7 (n = 6)*	*NS†, T = 10 (n = 6)*	*NS†, T = 3 (n = 4)*

Table 10.8. Comparison of learnability values between task types. Where sample size has been reduced for ties this is indicated, otherwise n = 8. † Marks failure of prediction.

Results of these tests indicate a high proportion of failures of prediction. However, as with the iconic interface study, it may be, because of the small sample size, that searching for statistically significant effects alone is not the most suitable way of testing the predictions. Rather, significance levels should be looked at in conjunction with mean performance values associated with each type of task. From these means it appears that task type A is more learnable than types B and C. Although time on task was the only measure of performance that produced statistically significant differences for both comparisons, the mean performance measures indicate that type A tasks take less than a third of the trials to learn, than types B and C do, in terms of errors. Similarly, in terms of help consultations, mean number of trials to learn type A tasks, is about half of the number required to learn task types B and C, yet, again, this does not show up as being statistically significant. A possible interpretation of these results, then, might be to accept that type A tasks are more learnable than either type B or type C tasks. However, because of the small sample size, the only performance measure sensitive enough to produce statistically significant differences was time on task.

Comparison of means for type C tasks with type D tasks, however, suggests that the lack of statistically significant differences may be a true reflection of the tasks' comparative learnabilities — thus marking real failures of prediction. Taken with the comparison of type A vs type B, this suggests that, in this context, rule compatibility may only aid learnability where there is also set compatibility. Indeed, in both previous studies where learnability was measured, with menu

based and iconic interfaces, no statistically significant differences were found between performance on task types C and D. A possible explanation is that the problems associated with a lack of set compatibility "swamp" any effects that a lack of rule compatibility might cause.

A subject performing a set incompatible task may go through two mental stages — firstly recalling the two possible command names associated with a particular operation, and then choosing between them. The predictions about the benefit of rule compatibility to learnability are based on the assumption that rule compatible command names could be recalled more efficiently. However, it might be that, in this context, the majority of subjects' time and effort was expended in making the choice between names, rather than in the recall — therefore any variances in the choice stage of the task may swamp rule compatibility effects. If a task was set compatible, however, subjects would not have to go through the "choosing" stage, making recall effort, and thus rule compatibility, the major factor — hence the clear differences between performance on task types A and B.

Consider, for example, the interface described by the formal notation in table 10.3. Here, the set incompatible experimental tasks are formatting a page (type C), and inserting a word (type D). If the ideas discussed above are correct, a user formatting a page will first recall the command names "FORMAT" and "STYLE", and will then try to decide which one is used when a page, rather than a graphic, is the object (a graphic is the object for the associated filler task). Meanwhile, a user inserting a word would recall the command names "BAZ" and "FOO", and then decide which was needed when a word rather than a letter was the object. The results of the three experiments where learnability was measured suggest that, by the time performance reached the criterion levels, the effect of having to choose between names was beginning to swamp effects of ease of recall. Note, however, that the results of comparison between type A and type B tasks (in all three studies), show that rule compatibility does affect learnability. Thus, it really does appear to be a case of effects being swamped, rather than being nonexistent.

The above discussion suggests that effects for set compatibility would be expected, whether or not rule compatibility was present. The results here indicate a clear effect where rule compatibility is present (A vs C), but the effect is less clear where rule compatibility is absent (B vs D). Although the WSR tests show that differences are not statistically significant, mean values do indicate that type B tasks may be more learnable. Type D tasks take 22%, 40%, and 12% more trials

to fulfil learnability criteria in terms of time on task, errors, and help consultations respectively. Further, the time on task figure is a lower bound on differences, as two subjects failed to satisfy the criterion for "learning" type D tasks, and were thus given the lower bound default value of 12 trials, whereas all reached criterion levels for type B during the session. Had this default value been set at, say, 15 trials then the mean learnability value associated with type D tasks would be 7.75 trials (35% more trials than type B), or had it been 20 mean learnability would be 9.00 trials (57% more). Further, statistically significant differences for the learnability of type B and Type D tasks was found in the other two studies where learnability was measured.

10.2.3 EUP

Performance over the later experimental trials is illustrated in figures 10.4 and 10.5.

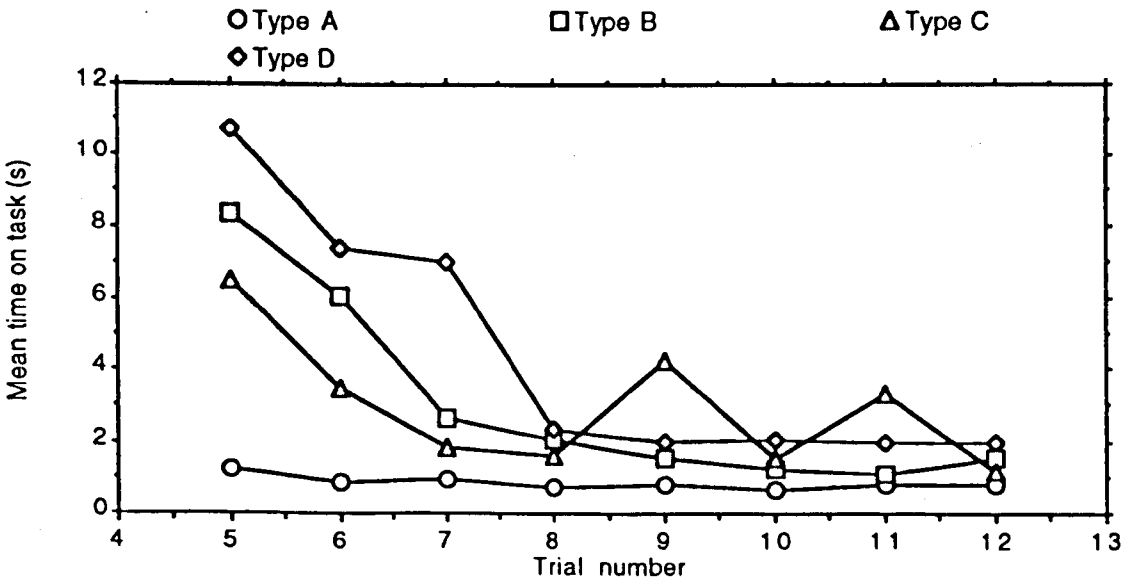


Figure 10.4. Mean time on task (seconds) on later trials for each task type.

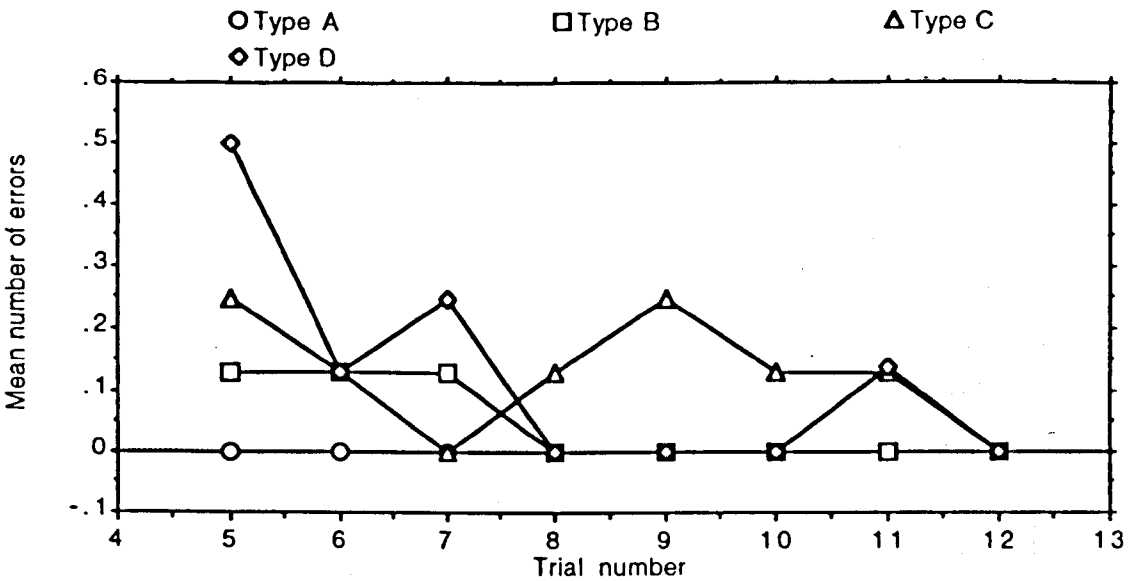


Figure 10.5. Mean number of errors on later trials for each task type.

Performance over the final three trials was taken as indicative of EUP. Mean EUP values are listed in table 10.9. Because of the small numbers of errors and help consultations at this stage, each subject's total errors and consultations over the last three trials were used in calculations. For time, figures given represent mean performance per trial. As with other studies a linear regression equation was fitted to the last four points of a mean plot of cumulative number of tasks vs. cumulative time. Values of R-squared were .998, .996, .967 and 1.000 for task types A, B, C, and D respectively, indicating that it was reasonable to assume that performance had levelled off by this stage. (Although the R-squared value associated with type C tasks was comparatively low, this was probably due to one subject's data distorting the result — she took over 16 seconds to complete a type C task on her eleventh trial).

MEASURE	Type A	Type B	Type C	Type D
Time	0.77 s (sd = 0.18 s)	1.34 s (sd = 0.54 s)	2.03 s (sd = 2.19)	2.14 s (sd = 0.74 s)
Errors	0.00 (sd = 0.00)	0.00 (sd = 0.00)	0.25 (sd = 0.46)	0.13 (sd = 0.35)
Help consultations	0.00 (sd = 0.00)	0.00 (sd = 0.00)	0.00 (sd = 0.00)	0.00 (sd = 0.00)

Table 10.9. Mean EUP values.

Only three subjects made errors, two on a type C task and one on type D. This is where it was expected errors would occur at EUP — on the two set incompatible types of task. However, it is not worth testing for statistical significance with so little data. No help consultations were made, however the time data was used in WSR tests of statistical significance. These results reinforce the point, made originally in chapter 4, about the comparative sensitivity of different measures. The time measures differentiate between performance on each type of task for every subject, error data only differentiates between one type of task and the other three types for three subjects, whilst help consultations shows no differences at all. Outcomes of the WSR tests are listed in table 10.10.

Comparison	Prediction	Time on task
A vs C	A > C	$p < 0.02$, T = 1
B vs D	B > D	$p < 0.05$, T = 3
A vs B	A = B	*†p < 0.02, T = 1*
C vs D	C = D	NS, T = 9

Table 10.10. Comparison of EUP values between task types (n = 8). †Marks failure of prediction.

As predicted, set compatibility affected EUP, regardless of the presence or absence of rule compatibility. Contrary to the predictions, however, there appears to be an effect for rule compatibility, where there is already set compatibility. Given the small sample size, finding statistical significance indicates that the effect may be particularly strong. This view is backed by the values listed in table 10.9, which show that subjects were taking, on average, almost twice as long to complete type B tasks as type A ones. So, once again, this marks a real failure of prediction, not simply, say, a swamped effect, and cannot be explained by the limitations of a small sample size.

A possible explanation is that performance hasn't yet started to level off. However, the performance curves illustrated in the figures appear to have "bottomed out" by the last three trials. Also, all subjects had stopped making errors on task types A and B by this stage — most long before. Further, the results of other comparisons suggest, both in terms of statistical significance and the mean values, that EUP has been reached. Perhaps, then, the deviation from the original predictions has arisen due to the nature of the task in this study — a recall rather then recognition based task.

If users group tasks as assumed, there will be only be one "mental" part to set compatible tasks — recalling the argument name for a particular operation (remember, the object argument was always as per the experimenter's instruction). So, why should type B tasks take longer than type A tasks here, when they didn't with the menu driven interface (noise in the data gained with the iconic interface made results difficult to interpret). The answer may be connected with the size of the problem spaces with which subjects were working, and the approach taken to solving the problem.

Norman (1988) makes a distinction between what he refers to as "knowledge in the head" and "knowledge in the world". He claims that both, or either, of these might potentially be harnessed for task completion, depending on the situation. When using a recognition based interface the user can utilise knowledge in the world or, in this case, knowledge in the interface. With the menu driven interface, users knew they must open one of the seven menus displayed. The task, then, may have taken the form of looking at the choices available, picking the menu which they recognised as being associated with a particular set of commands and then opening it in order to invoke the command requested by the experimenter. The users were, then, using the interface to provide them with action cues (the menu headings). Also, the problem space was small — the users having to choose between only seven alternatives. Thus, the a priori "naturalness" of the command menu heading may be of little consequence, the important issue being that there be a single heading per command set (otherwise users would have to "prise apart" the members of a set in their heads, before they can start a task).

With recall based tasks, the user cannot use the interface for cues, but rather will be dependent on knowledge in the head. This means the problem space may potentially include a user's entire mental contents — a huge space when compared to the seven menu titles. In such situations, then, it may not be surprising if factors which can aid a user's mental search — such as the a priori naturalness of a command name — do have a significant effect on performance. Even with this type of interface, however, the effects of rule compatibility seem far smaller than those for set compatibility. Performance on type C tasks was no better than on type D, and the performance gap, in terms of time, between type C and type A tasks was over twice that between type B and type A tasks.

10.2.4 System potential

This measure is of little interest in the context of this study. It should be the same for all types of task in terms of errors and help consultations (0). In terms of time on task there may be small differences, due to search times associated with finding different keys. However, as the allocation of task to task type was balanced between the four experimental interfaces, it was not anticipated that there would be any systematic effects.

10.2.5 Re-usability

Values for re-usability were taken from subjects' performance in the trial completed in the second experimental session. As there were no errors, and no help consultations were made, results are given in terms of time only (table 10.11). One subject's data was excluded from the analysis, due to experimenter errors in task setting.

MEASURE	Type A	Type B	Type C	Type D
Time	0.89 s (sd = 0.32 s)	3.51 s (sd = 1.53 s)	2.06 s (sd = 1.01 s)	2.32 s (sd = 2.32 s)

Table 10.11. Mean re-usability values (n = 7).

Results of WSR tests for statistically significant differences are given in table 10.12. Note that the exclusion of one subject's data means that the sample size is seven.

Comparison	Prediction	Time
A vs C	A > C	$p < 0.05$, T = 1
B vs D	B > D	*†NS, T = 5*
A vs B	A > B	$p < 0.02$, T = 0
C vs D	C > D	*†NS, T = 11*

10.11. Comparison of re-usability values (n = 7). †Marks failure of prediction.

Comparison of type A tasks against types B and C showed effects for both set and rule compatibility as expected. However, comparison of type D tasks with types B and C showed no similar effects. These two comparisons also failed to show

differences in the menu based study reported in chapter 7.

This seems difficult to explain. A possibility though, is that performance has been affected by subjects' behaviour whilst away from the interface. Subjects had been told that they would be asked to complete a further trial in their second session, and it is possible, therefore, that they may have mentally rehearsed before the trial started. As type D tasks were apparently the most difficult, these may have received special attention.

The methodology used here for measuring re-usability might also be somewhat questionable. Possible effects due to lack of concentration, or lack of motivation to complete the task quickly, at the beginning of the session may have had an effect. In the context of measuring guessability, where task completion times are comparatively long, it may be reasonable to expect these effects to be rendered insignificant. However, with re-usability data, where times are comparatively short, this could be a significant source of noise.

10.3 SUMMARY

A study has been conducted looking at the effects of set and rule compatibility in the context of performing tasks with a command driven interface. Broadly, effects were as predicted — rule compatibility affecting early performance, with set compatibility having a greater effect later on. Studies testing the predictions made about the effects of consistency on usability have now been conducted using three different types of experimental interface — text menu based, iconic, and command driven. Overall, the indications are that the effects of consistency are fairly general across these.

However, some failures of prediction were found in this study. In particular, effects due to rule compatibility seem to remain for longer that they did with the menu based interface. This may be partly attributable to differences between recall and recognition based tasks — in particular the comparative sizes of problem space.

Chapter 11

INVESTIGATING THE EFFECTS OF CONSISTENCY AT A HIGHER LEVEL OF AN INTERFACE, WITH HIGHER LEVEL TASKS

In this chapter the notion of levels of task and levels of an interface are introduced and discussed. The level of task set is likely to affect the way in which the associations influencing consistency are made, as well as, possibly, the degree of inter-subject variability. Consistency can also be considered at different levels of an interface.

The consistency studies reported so far in this thesis have been in the context of three different types of interface — menu based, iconic, and command driven. However, as will be explained, neither the level of the tasks set, nor the level of the interface at which inconsistencies have been introduced, has been varied.

In the next two sections, the concepts of task levels and interface levels are introduced, and their possible influence on the perception and effects of inconsistencies discussed. A study is then reported, investigating whether the predictions about the effects of consistency on usability can be generalised to higher level tasks and higher levels of the interface.

11.1 LEVELS OF TASK

Tasks can exist at different levels of complexity. Consider Draper's (1993) example of a word processor operator producing a letter. This is a fairly high level task, which will contain several lower level tasks, perhaps typing text, or some formatting tasks. The higher the level of a task the less tightly defined it is likely to be. This will probably mean, firstly that there will be a wider inter-user variation in the way the task is approached than there would be in a lower level task, and secondly that it might be more difficult to predict effects that certain properties of an interface may have on the efficiency with which the task can be performed.

Consider, again, the task of producing a letter. This task might mean different things to different groups of users. Secretaries, for example, might put a lot of emphasis on the appearance of their letters — thus, for them, low level tasks might include formatting and page layout. Conversely, a person communicating with a friend might not bother with such things. Because of this inter-subject variability, it might be difficult to make general statements about the overall effects which specific aspects of a design might have on usability for this task. This issue is central to Bannon and Bodker's (1991) objections to the application of task analyses to high level tasks.

However, as Suchman (1987) noticed, even if a task is tightly defined in the way it is presented to users, there may still be room for variation in the actions which users take to complete the task. If the contents and formatting of a letter were clearly defined, it might still be possible for different users to achieve the required outcome by different methods. For example, one user might decide to type in all the text first, and then set margins and font styles later, whilst another might decide to format the letter as he or she went along. This was illustrated by Allen and Scerbo's (1983) study, in which they compared predictions from the GOMS task analysis method to experimental findings. They found that, even with well defined tasks, unless the exact method was dictated to the subject, performance could not be reliably predicted. As Draper (1993) points out, dictating exact methods to subjects is most unnatural, and is certainly not likely to reflect practice outside of the laboratory. With lower level tasks, however, there is likely to be less room for variation in the action-rules employed.

In the consistency experiments reported so far in this thesis, the tasks were all low level — the invocation of commands at the request of the experimenter. There was, therefore, comparatively little room for variation in approach to these tasks. A research question still to be addressed, then, is whether effects for consistency, common to a user population, could be found with more complex tasks, or would its effects be lost in the noise of inter-subject variability.

Another issue, connected with the type of task set to subjects, is the level at which the associations which determine consistency are formed. In the consistency studies reported so far, subjects have simply been given the name of a command, and asked to take the necessary actions to invoke it. The subjects in these experiments may not have had an understanding of the outcomes of command invocation, so consistency issues may have simply been to do with word-word, or

word-graphic associations, rather than associations made because of the nature of the operations performed. For example, in the experiment with the command driven interface (reported in the previous chapter), two of the tasks which the subjects were asked to do were to invoke the command for deleting a table, and the command for deleting a word. Although subjects may have had ideas about what the outcome of performing such tasks would be in a working package, it is also possible that the only reason they were inclined to treat these tasks as similar was that, for each, the experimenter had asked them to invoke a command for "deleting".

Had the tasks been set at a higher level, for example, in terms of achieving particular effects, subjects may not necessarily have thought of the tasks as similar. Another possibility, is that subjects may have thought of these tasks as similar but not as "deleting" tasks. For example, Macintosh users might think of them both as "cutting" tasks (from the "cut and paste" metaphor). Had the instructions been given in a more neutral manner (i.e. without mentioning the word "deleting"), then they might have expected the associated operation argument to be "cut".

Similarly, if deleting text and tables had been sub-tasks in a higher level task, associations could be made on different bases. For example, if a user decided to delete text and tables as elements of the task of preparing an academic paper, then he or she might regard the tasks as similar. Conversely, if one were part of, say, writing a letter, and the other part of presenting statistics, users might treat them as being different.

11.2 LEVELS OF INTERACTION

There is more than one level at which a user can interact with an interface. For example, Moran (1981) divides interaction into the following six levels: task level, semantic level, syntactic level, interaction level, spatial layout level, and device level. Another factor which was common to the consistency studies reported so far, was the level of the interface at which inconsistencies were investigated. In these, consistency was concerned with the elements in the rule schemata, rather than in the number and types of actions included in the schemata. The latter, then, would represent consistency at a different level. For example, consider the tasks of invoking the commands for, say, putting text into italics and putting text into

New York style. Assume that a person regards these tasks as being similar. Imagine that this person is a Microsoft Word user with an Apple Macintosh computer, and so needs to employ the following rules for the two tasks:

to put text into italics -> highlight text + open FORMAT menu + select ITALIC
to put text into New York style -> highlight text + open FONT menu + select NEW YORK

On one level these tasks are not set compatible — one requires the user to open the FORMAT menu, and the other the FONT menu. They cannot be represented by a single schema of the form of, say:

to put text into style X -> highlight text + open FORMAT menu + select X

where X = [ITALIC, NEW YORK]

On another level however, the two rules are similar. Both have three elements — highlighting text, opening a menu, and choosing a command. They could thus be generalised into a single higher level rule, such as

to put text into style X -> highlight text + open menu Y + select X

where X = [ITALIC, NEW YORK], and Y = [FORMAT (X = ITALIC), FONT (X = NEW YORK)]

This higher level rule, then, has captured a similarity which would have been missed by the former notation, which, however, has highlighted the lower level inconsistencies.

Of course the lower level inconsistencies are likely to cause the user problems — this has been demonstrated in the studies reported so far. It seems likely that users might, say, open the 'FONT" menu when trying to put text into italics. Equally, however, there may be gains from the higher level similarities. For example, having failed to find the "ITALIC" command on the "FONT" menu, it seems likely that the user might look on other menus. This strategy should ultimately prove fruitful, assuming the user tries the "FORMAT" menu at some stage. Had this higher level regularity not existed, then the user might have had more difficulties. For example, if the rule for putting text into italics were, say:

to put text into italics -> highlight text + hold down command key + type "I" + press return,

then there seems no reason to expect any useful transfer from putting text into New York style, beyond the highlighting part of the task. In this case, then, it would not be possible to generalise the two rules, even at the higher of the levels suggested so far. However, if prepared to go to a still higher level, perhaps some generalisation could still be attempted, capturing the need to highlight the text to be operated on. This might take the form of say:

to put text into style X -> highlight text + perform command invocation sequence Y

where X = [ITALIC, NEW YORK], and Y = [hold down command key + type "I" + press return (X = Italic), open FONT menu + select NEW YORK (X = NEW YORK)].

The point is, then, that consistencies can be perceived at different levels of an interface, and may still be of some use. The higher the level of perception the less specific the transferable parts of the rule. In the first examples in this section, consistency was being considered in terms of similar commands being contained in the same menu, yet the last example has illustrated consistency at the level of highlighting text and then going through an appropriate command sequence. Of course, consistencies may be more difficult to perceive on higher levels, and could also, because of the less specific nature of the links, be less useful to the user. In the last example given, the regularities may lead the user to highlight text and search for a command invocation sequence, but there seems little reason to expect that knowledge of the sequence needed to put text into New York style would be of much use to someone wanting to put text into italics. At least, however, users aren't required to, say, change the mode of the word processor, or change to another application package.

All the examples so far have been concerned with set compatibility, but what about rule compatibility? Again, this can exist on several levels. Say, that a user of Microsoft Word for the Macintosh expected that in order to save their work they had to choose a command "SAVE" from the "UTILITIES" menu. The expected rule would be:

to save -> open UTILITIES menu + choose SAVE.

In fact the rule required is:

to save -> open FILE menu + choose SAVE.

Clearly, on one level this is rule incompatible — the user expects to find the command on the "UTILITIES" menu, but it isn't there. However, on another level there is rule compatibility. The user expects to have to open a menu and select a command, which indeed is the case. Had the user expected to have to type in a command, then there would not even have been rule compatibility on the higher of these levels.

11.3 THE STUDY

The reported study was designed to investigate the effects of consistency at a higher level of the interface than for the earlier studies, and with a higher level of task.

Consistency, in this study was concerned with the method by which subjects interacted with the experimental interface — via menus or a command key, depending on the task to be performed. Tasks were set in terms of asking subjects to achieve particular effects. So, for example, instead of simply being asked to invoke a command called Bold, subjects would be asked to put a section of text into bold style writing. They would also be shown what bold text looked like, so that they would understand the effect of their actions. Although the experimental interface was designed so that a specific action-rule was needed for each task, the way in which tasks were set — making sure that subjects had an understanding of what they were being asked to do — meant that user's inter-task associations and a priori expectations about action rules, would not be formed solely on the basis of command names.

As before, subjects were set four types of task reflecting different combinations of set and rule compatibility (see table 11.1).

	RULE COMPATIBLE	RULE INCOMPATIBLE
SET COMPATIBLE	Type A	Type B
SET INCOMPATIBLE	Type C	Type D

Table 11.1. Set and rule compatibility of experimental tasks.

The experiment was designed to test hypotheses based on the predictions made about the effects of set and rule compatibility on guessability for subjects with and without previous experience of the interface for similar tasks. The hypotheses tested are listed in table 11.2.

EXPERIENCE WITH THE INTERFACE FOR SIMILAR TASKS	EFFECT OF SET COMPATIBILITY	EFFECT OF RULE COMPATIBILITY
No previous experience	A = C, B = D	A > B, C > D
High previous experience	A > C, B > D	A = B, C = D*

Table 11.2. Experimental hypotheses. *Note that this prediction differs from those made in the studies reported in chapters 8 and 9. The reason for this is explained later, in the "Results" section.

11.3.1 Apparatus

Microsoft Word 4 for the Apple Macintosh was specially adapted for the study. Four separate interfaces were designed, the intention being that each should support tasks of types A, B, C, and D, and so allow a within subjects experimental design. There were two mechanisms for command activation with these interfaces — picking commands from menus, or using the command ("apple") key in conjunction with others. Set and rule compatibility of the experimental tasks was to be manipulated according to which activation method was used for activating the commands associated with these and associated filler tasks. So, for example, if text could be put into bold style by picking a command from a menu, yet in order to put text into italic style users had to use the command key, these tasks would be set incompatible. Pre-experimental training was designed to make using menus more rule compatible than using command keys. (This is detailed in the experimental design section).

The "COMMANDS" feature on the "EDIT" menu enables commands to be removed from, or added to, menus It also allows command key combinations for activating tasks to be specified. For each interface, these were altered from the default settings for some of the commands, in order to manipulate the consistency of the experimental tasks. Aside from these changes the interfaces were left in the default setting. Subjects were to be set fourteen tasks which, it was anticipated, they would divide into four separate sets. Two of the tasks in each set were to be treated as experimental tasks. Table 11.3 lists formalised task action rules for one of the interfaces. The experimental tasks are shown in italics, the rest — in the experimental context — being fillers. All tasks were set in the context of editing and formatting letters.

Taskset
TASKSET = [*move text up to the right hand margin*, move text to centre of page, *move text up to the left hand margin, put text into bold style writing*, put text into italic style writing, put text into outline style writing, *put text into shadow style writing, put the date into the letter*, put page numbers into the letter, *put the current time into the letter, move the cursor to the end of the letter*, move the cursor to the start of a letter, move the cursor to the start of a line, *move the cursor to the end of a line*]

Schema sets (agent = designer)
SET U = [*move text up to the right hand margin*, move text to centre of page, *move text up to the left hand margin, put the date into the letter*, put page numbers into the letter, *move the cursor to the end of a line*]

SET V = [*put text into bold style writing*, put text into italic style writing, put text into outline style writing, *put text into shadow style writing, put the current time into the letter, move the cursor to the end of the letter*, move the cursor to the start of a letter, move the cursor to the start of a line]

Rule schemata (agent = designer)
to perform task U -> highlight text + Open appropriate menu + select appropriate command

to perform task V -> highlight text + hold down "apple" key + hold down "ctrl" key + hold down appropriate letter key

Schema sets (anticipated) (agent = user)
SET P = [*move text up to the right hand margin*, move text to centre of page, *move text up to the left hand margin*]

SET Q = [*put text into bold style writing*, put text into italic style writing, put text into outline style writing, *put text into shadow style writing*]

SET S = [*put the date into the letter*, put page numbers into the letter, *put the current time into the letter*]

SET R = [*move the cursor to the end of the letter*, move the cursor to the start of a letter, move the cursor to the start of a line, *move the cursor to the end of a line*]

Rule schemata (anticipated) (agent = user)
to perform task P -> highlight text + open appropriate menu + select appropriate command
to perform task Q -> highlight text + open appropriate menu + select appropriate command
to perform task R -> highlight text + open appropriate menu + select appropriate command
to perform task S -> highlight text + open appropriate menu + select appropriate command

Set compatibility
SET P is a sub-set of SET U (set compatible)
SET Q is a sub-set of SET V (set compatible)
SET R is spread across SET U and SET V (set incompatible)
SET S is spread across SET U and SET V (set incompatible)

Rules match for: *move text up to the right hand margin*, move text to centre of page, *move text up to the left hand margin, put the date into the letter*, put page numbers into the letter, *move the cursor to the end of a line* (rule compatible)

Rules do not match for: *put text into bold style writing*, put text into italic style writing, put text into outline style writing, *put text into shadow style writing, put the current time into the letter, move the cursor to the end of the letter*, move the cursor to the start of a letter, move the cursor to the start of a line (rule incompatible)

Table 11.3. Formalisation of task action rules associated with an experimental interface.

Notice that the rule schemata have been outlined in far less specific terms than in formalisations given in previous chapters. However, what is given here is sufficient to illustrate where inconsistencies occur. Another difference between this and previous interface formalisations, is that the designer's task grouping has been given as consisting of two large sets as opposed to the users' four smaller ones. Thus instead of looking for matches between sets, set compatibility can be identified by whether a user's schema set is a sub-set of just one of the designer's sets. If it is then the designer is treating tasks, which the user regards as being in the same set, as being similar. Experimental tasks by task type for the interface described are given in the "Experimental design" section.

Experimental tasks were balanced for task type (A, B, C, or D) across the four interfaces using a Latin square design. So, if, say, with one interface putting text into bold style writing was a type A task in one interface, it would be type B in another etc.

Help files were available for subjects to consult. The experimenter opened these before the experiment. This meant the names of these files would be listed on the "WINDOW" menu, and that the files could be accessed in the same way that a menu based command would be invoked. The file names all started with "Help" followed by the name of the command to which they referred (names mirrored menu command names, or, if the command had to be activated via the "apple" key, the name that the command would have had, had it been on a menu).

11.3.2 Subjects

Sixteen students from the University of Glasgow participated in the study. The subjects — thirteen females and three males — all had previous computing experience, and seven had experience of other word processing packages. However, none had used an Apple Macintosh before, nor used Microsoft Word in another operating environment. Subjects were recruited on a "first come first served" basis from the Psychology Department's undergraduate laboratory. The only criterion for eligibility was having no previous experience of a Macintosh computer. Three pounds was paid for participation.

11.3.3 Experimental design

The intention was to obtain values for two types of guessability — guessability with no previous experience of the interface, and guessability with previous experience of similar tasks within the same interface. Measures for the first were taken by setting an experimental trial including tasks of types A, B, C, and D, without giving subjects prior opportunity to practice with the interface. After this trial subjects were then set a series of filler tasks, before a second experimental trial was set. This trial again contained four tasks, one of each type. New tasks were used in the second experimental trial. There were seven blocks of fillers, containing four tasks each, between the two experimental trials. Each block contained tasks which, it was assumed, subjects would regard as similar to the tasks in the second experimental trial. The tasks set in the first experimental trial also acted as fillers for the second experimental trial, and these reappeared during subsequent filler blocks, along with the other filler tasks set. Excluding the tasks set during the first experimental trial, there were either one or two different fillers associated with each of the tasks in the final experimental trial (depending on the amount of (apparently) similar functionality available in the interface). The tasks set during each trial and filler block are listed in table 11.4. The letters in parentheses refer to subjects' anticipated settings (as per table 11.3).

Trial number / Filler block	**Tasks set**
Experimental trial 1	move text up to the right hand margin (P) put text into bold style writing (Q) move the cursor to the end of the letter (R) put the date into the letter (S)
Filler block 1	move text to the middle of the page (P) put text into italic style writing (Q) move the cursor to the beginning a line (R) put page numbers into the letter (S)
Filler block 2	As for experimental trial 1
Filler block 3	move text to the middle of the page (P) put text into outline style writing (Q) move the cursor to the beginning of the letter (R) put page numbers into the letter (S)
Filler block 4	As for experimental trial 1
Filler block 5	As for filler block 1
Filler block 6	As for experimental trial 1
Filler block 7	As for filler block 3
Experimental trial 2	move text up to the left hand margin (P) put text into shadow style writing (Q) move the cursor to the end of a line (R) put the current time into the letter (S)

Table 11.4. Tasks set in each trial.

In this context, then, the experimental tasks were of the following types, listed in table 11.5, when performed using the interface described in table 11.3.

Experimental trial number	Type A	Type B	Type C	Type D
1	move text up to the right hand margin	put text into bold style writing	put the date into the letter	move the cursor to the end of the letter
2	move text up to the left hand margin	put text into shadow style writing	move the cursor to the end of a line	put the current time into the letter

Table 11.5. Experimental tasks by tasks type.

The tasks were set in the context of editing and formatting a series of different letters. An example of a letter before and after the experimental session is given in appendix 11.1.

A measure of performance used was whether or not subjects' initial attempts at command invocation were made via the menus, or the "apple" key. If subjects attempted to interact via the appropriate method, this was noted regardless of whether they actually managed to activate the command. So, for example with reference to the interface described in table 11.3, if a user's first attempt at moving text to the right had margin involved opening, say, the "EDIT" menu to try to find the appropriate command, his or her method would be regarded as correct, even though the command needed is on a different menu ("FORMAT"). This was possibly the most important usability measure employed in this study, as it was directly concerned with the level of interaction at which inconsistency was introduced. Other performance measures taken were: time on task, total number of errors (including errors such as opening the wrong menu, or trying the wrong command key combination, or invoking an inappropriate command), and whether or not the help files were consulted.

Time on task and total number of errors were, perhaps, less "pure" measures as they were likely to be contaminated by effects which were not central to the purpose of the study. With the tasks requiring menu command activation, for example, errors due to subjects attempting to open incorrect menus would affect these measures, yet were not directly what the experiment was concerned with. Similarly, there would be a penalty for those trying inappropriate command key

combinations, even if command keys were required for the task. Both of these factors were also likely to affect the time on task measure.

11.3.4 Procedure

After reading an overview of the experiment (appendix 11.2) and completing a personal details interview, subjects were introduced to the principles of the Macintosh operating environment. This was done by talking subjects through a number of practice tasks using the Statview statistics package. The tasks set involved the subjects invoking commands, both from the menus, and using the "apple" key in combination with others from the keyboard. Aside from teaching subjects the "mechanics" of command invocation, the main purpose of the practice session was to influence the users' mental contents prior to the experimental session.

Subjects were told that when working in the Macintosh operating environment, task invocation would either be by picking commands from menus, or by using the "apple" key in combination with others, depending on the task they were set. However, of the ten practice tasks eight required menu use, but only two the "apple" key. The aim, then, was to leave subjects with the impression that menu use was the norm — thus influencing expectations during the experimental session, and making menu based tasks more rule compatible. Tasks set during the practice session are listed in appendix 11.3.

When the practice session was over, the experimental session started. The first experimental trial started straight away, and there were no gaps between tasks right through until the end of the second experimental trial. Instructions were given in the form of the experimenter explaining the effect the subjects were to achieve, rather than by requests to invoke specific commands. For example, if the task was to move the cursor to the end of a letter, the instruction would be "please could you move the cursor to the end of the letter", rather than, "please could you invoke the command "Move to end of document"". The experimenter might also point to the place on the screen which the cursor was to moved to. Any questions relating to what the subjects had been asked to do were answered, and experimenter-subject dialogue continued, until both agreed that the subject knew what he or she was being asked to achieve.

Subjects were advised that they could use the help system whenever they wished, but, equally, that they should not feel reticent about exploring the menus, or possible "apple" key combinations. Indeed, they were advised that they should not worry about making errors, and that, therefore, it might sometimes be worth making a guess.

The experimenter timed subjects from when he had finished explaining each experimental task, to when the task had been successfully completed. Any errors or help consultations were also noted. The experimenter followed a written protocol throughout the experimental session (appendix 11.4), which lasted for about forty-five minutes per subject.

11.4 RESULTS AND DISCUSSION

Performance on the first and second experimental trials is illustrated in terms of the various performance measures in figures 11.1, 11.2, 11.3, and 11.4.

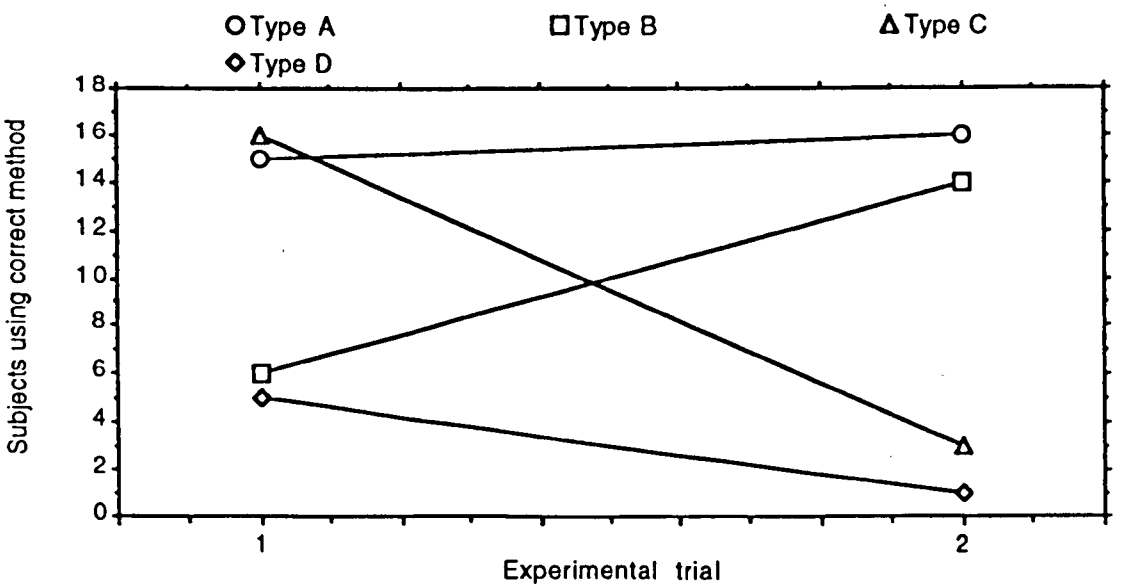


Figure 11.1. Number of subjects using the correct interaction method for each task type.

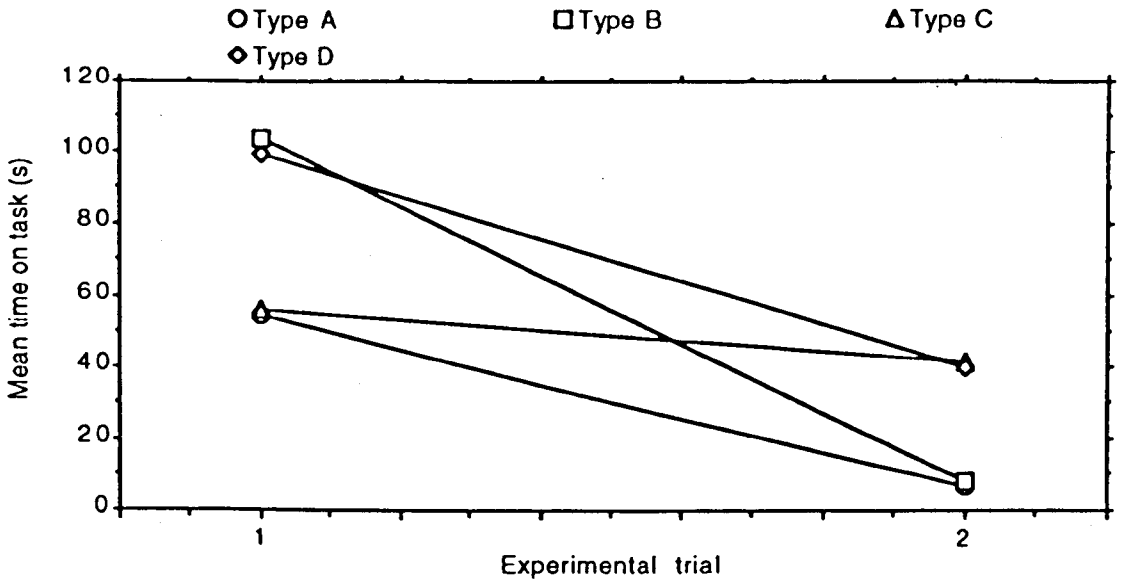


Figure 11.2. Mean time on task (seconds) for each task type.

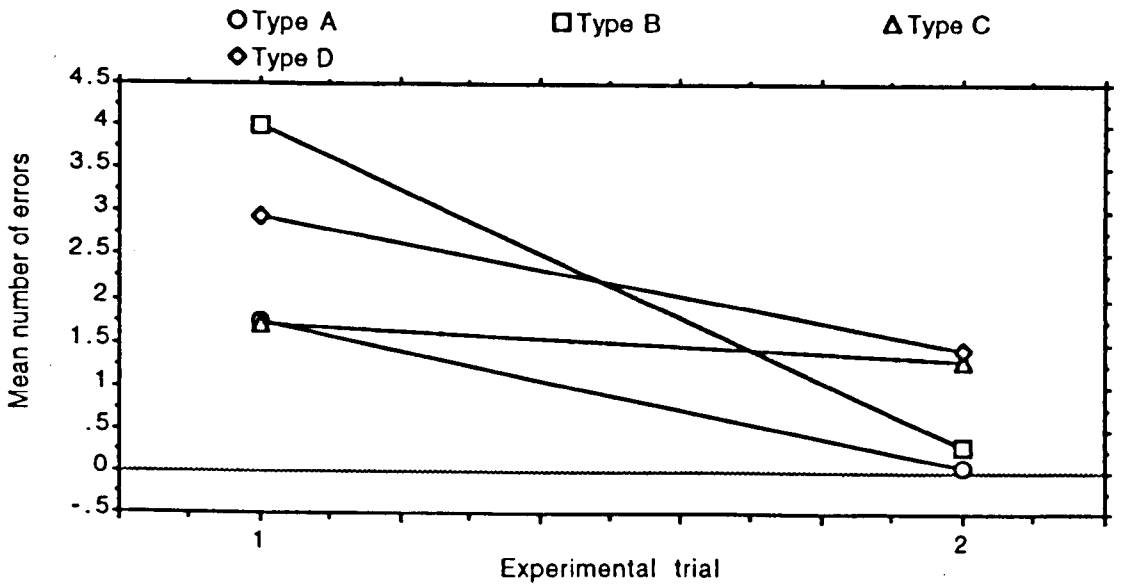


Figure 11.3. Mean number of errors for each task type.

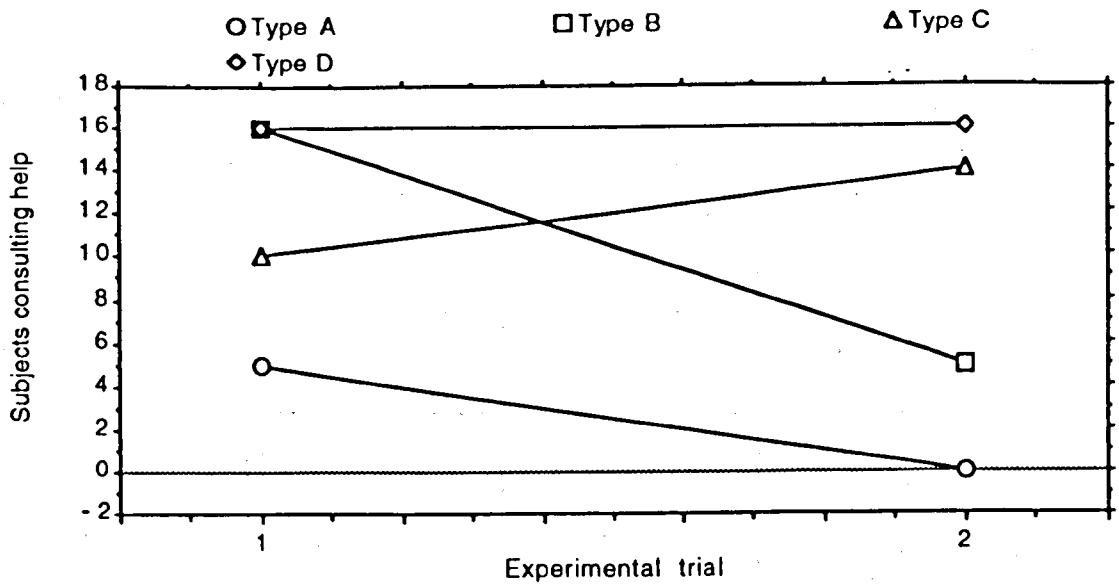


Figure 11.4. Mean number of subjects consulting help for each task type.

11.4.1 Guessability without previous experience

Performance on the first experimental trial was taken as being indicative of guessability for those without previous experience of the interface. Results are summarised in table 11.6.

Task type	Number using correct interaction method	Mean time on task	Mean of total number of errors	Number consulting help files
A	15 (94 %)	54.1 s (sd = 31.7 s)	1.75 (sd = 1.65)	5 (31 %)
B	6 (37 %)	103.5 s (sd = 53.7 s)	4.00 (sd = 3.86)	16 (100 %)
C	16 (100 %)	56.1 s (sd = 32.9 s)	1.69 (sd = 2.33)	10 (63 %)
D	5 (31 %)	98.9 s (sd = 49.5 s)	2.94 (sd = 2.52)	16 (100 %)

Table 11.6. Mean guessability values for those without previous experience of the interface (n = 16).

A WSR test was carried out to check for statistically significant effects in terms of time on task and total errors. As expected highly significant effects were found for rule compatibility (see table 11.7).

Comparison	Prediction	Time on task	Total errors
A vs C	A = C	NS, T = 66	NS, T = 34, n = 12
B vs D	B = D	NS, T = 67	NS, T = 25, n = 12
A vs B	A > B	$p < 0.02$, T = 22	$p < 0.05$, T = 10.5, n = 12
C vs D	C > D	$p < 0.02$, T = 20	$p < 0.10$*, T = 15, n =12

Table 11.7. Comparison of guessability values for those without previous experience of the interface (where tied values give $n < 16$, this is indicated).

These results appear to strongly support the predictions about the effects of set and rule compatibility on guessability for those without previous experience of the interface. Perhaps the only surprise is that as many as six and five subjects respectively tried command key entry first with task types B and D respectively. However, the reason for this was probably that some of the subjects consulted the help files before their first attempt at a task — so for them performance would not necessarily reflect a priori ideas about which was the more natural rule.

However, some caution should be used in interpreting these results. Of the four performance measures used, three — time on task, total errors, and help consultations — seem prone to influence from factors other than set and rule compatibility at the level at which it is being investigated. With command key invocations, for example, subjects will suffer a penalty if they attempt the wrong key combination. Indeed problems are likely here as during the practice session subjects' command key activations did not require use of the "ctrl" key, and it seems unlikely that they could have anticipated the need to use it here. Similarly, with the menu commands, subjects may not have been able to anticipate where a command was placed. Again, the practice session would not have been helpful in this respect, as the different type of application — a statistics package — was used, and so subjects came across neither the same menu headings nor command names as those in the experimental interface. Whilst some, or possibly all, of these

issues may influence set and rule compatibility at lower levels of the interface, they are not concerned with consistency at the level which this study was designed to address.

The most useful measure here, then, might be the interaction method attempted, as this addresses performance at the level at which inconsistencies were introduced. This also lends strong support to the predictions — with all subjects attempting the correct method with type C tasks, and all but one with type A, as opposed to only six and five with task types B and D respectively.

11.4.2 Guessability with previous experience of the interface

Results are summarised in table 11.8.

Task type	Number using correct interaction method	Mean time on task	Mean of total number of errors	Number consulting help files
A	16 (100 %)	6.4 s (sd = 2.4 s)	0.06 (sd = 0.25)	0 (0 %)
B	14 (88 %)	8.2 s (sd = 8.6 s)	0.31 (sd = 0.87)	5 (31 %)
C	3 (9 %)	41.6 s (sd = 13.1 s)	1.31 (sd = 1.14)	14 (88 %)
D	1 (6 %)	40.6 s (sd = 15.3 s)	1.44 (sd = 1.15)	16 (100 %)

Table 11.8. Mean guessability values for those with previous experience similar tasks with the interface (n = 16).

Again, a WSR test was carried out to check for statistically significant effects in terms of time on task and total errors. This time highly significant effects were found for set compatibility (see table 11.9).

Comparison	Prediction	Time on task	Total errors
A vs C	A > C	$p < 0.002$, $T = 0$	$p < 0.002$, $T = 0$, $n = 14$
B vs D	B > D	$p < 0.002$, $T = 0$	$p < 0.02$, $T = 17.5$, $n = 15$
A vs B	A = B	NS, $T = 65$	NS, $T = 1$, $n = 3$
C vs D*	C = D	NS, $T = 61$	NS, $T = 8$, $n = 6$

Table 11.9. Comparison of guessability values for those with previous experience of similar tasks with the interface (where tied values give $n < 16$, this is indicated).

Again, these results appear to give strong support to the predictions, both in terms of mean values, and statistical significance. Whilst method of interaction may still be the purest measure in reflecting effects from set and rule compatibility, it may now be possible to treat results from the other measures with less caution. By this stage, users would have had a clearer idea both of the command key combinations required ("apple" key, "ctrl" key, and letter), and of the menu groupings (e.g. commands related to "BOLD" are on the "FORMAT" menu). This should reduce the noise generated from that type of error. Anyhow, each of the performance measures now indicates that it is set compatibility which is the more important factor here.

A prediction here (marked *), different to those relating to previous studies, is that rule compatibility would not be an advantage where there is an absence of set compatibility. Previously, it had been expected that this would be helpful when subjects made their second attempt at a task. The reason that no advantage was expected here was due to the size of the problem space. At the level at which consistency was manipulated, subjects only had two options — invocation via menus, or invocation via command keys. Thus if the first method were wrong, there would only be one alternative anyway. This contrasts to other studies where, for example, a subject who opened an unsuitable menu might have three others to choose from, and might make this choice by reverting to a priori expectations.

Although, statistically significant differences were not found, overall results indicate better performance with type A tasks than with type B, particularly in terms of help consultations. Although these differences are negligible in comparison with those between type A tasks and tasks types C and D, it could be that subjects' level of experience with previous tasks was not yet quite enough to

completely eradicate effects for rule compatibility. Had more fillers been set these differences should disappear.

11.5 SUMMARY

A study was designed to test the predictions about the effects of consistency on usability with a higher level of task, with consistency being manipulated at a higher level of the interface. This meant that inter-task associations and a priori action-rule expectations could be made on the basis of the nature of a task, rather than on the basis of word-word, or word-picture associations. Also, inconsistencies occurred due to different types of rule, rather than merely through different elements being included in rules which otherwise were similar.

An experiment investigated guessability for users with no previous experience of an interface, and guessability for those with experience of similar tasks. Results were as expected — rule compatibility being important in the former case, and set compatibility in the latter. This indicates that the predictions may be generalised to higher levels of task, and to situations where inconsistencies exist at higher levels of an interface.

Chapter 12

DOES CONSISTENCY MATTER IN "REAL" INTERFACES?

A stricture against which all HCI research should be tested, is that effects found in artificial laboratory studies may not translate into "real-life" contexts. For example, Landauer (1987) argued, on the basis of his own extensive work, that theoretically motivated effects demonstrated in the laboratory are sometimes insignificant in practical contexts. This is a criticism to which consistency research appears particularly vulnerable. Whilst experiments conducted for this thesis, and those reported from the literature, give strong support to the idea that inconsistencies can be detrimental to the usability of an interface, it is noticeable that all of these studies have been conducted under tightly controlled laboratory conditions. Furthermore, the experimental interfaces have all been created with inconsistencies deliberately present for the sake of investigation, and some of the tasks set to users appear somewhat artificial. In fact, with the exception of the experiment reported in the previous chapter, the studies reported in the thesis are unrealistic in at least three ways: users were hired subjects who would not have had the motivation of hoping to learn useful skills; the tasks performed were very artificial — e.g. text was not visibly edited; and the interfaces were designed by the investigator to show effects, not by a designer trying to be reasonably helpful to the user.

In this chapter the approach expounded throughout the thesis is applied to a commercially available word processor in the context of realistic tasks. Possible rule and set incompatibilities are identified which, it is predicted, will cause users difficulty. Data from a previously conducted, ecologically valid, usability investigation of this word processor (Microsoft Word 4.0 for the Apple Macintosh) (Jordan 1992) is then re-analysed to see whether these predicted difficulties had a significant effect on performance.

The aims of the original study were to make an overall statement about the guessability of the interface, and to identify potential problems that users might have during early interactions. No particular attempt was made to highlight the effects of inconsistencies, and thus there had been no manipulation of the interface to this end. Further, the experimental tasks and subjects were chosen to reflect a "real life" context of use.

The aim of the re-analysis was to address two research questions :

1. Do designers really build inconsistent interfaces? — Perhaps it is natural to design consistent interfaces. Inconsistencies may only occur if a designer is deliberately perverse.

2. Even if there are inconsistencies in an interface do they really matter? — Do the effects found in the laboratory transfer to "real life", or are they rendered insignificant by other factors?

The re-analysis involved checking the interface for inconsistencies, and then referring back to the original experimental data, to see if any of the users' problems may have been attributable to these.

As users' completion of the experimental task — which contained twelve sub-tasks — represented their earliest interactions with the interface, it was the guessability of each sub-task which was being investigated. Thus, it was anticipated that both set and rule incompatibilities could affect performance, the degree of the influence of each being dependent upon whether a user had previously completed a sub-task in the same set (from the user's point of view).

12.1 THE ORIGINAL STUDY

12.1.1 Subjects

The subjects of the study were twelve female secretaries, hired from a temporary employment agency. The subjects, whose ages ranged from from 17 to 41 years, all used word processors regularly (between 1 and 5 days per working week), and were all skilled typists. However, none had used Microsoft Word before, and only one had any experience of using an Apple Macintosh computer.

12.1.2 Experimental design

The task was to convert a letter from a "plain" form, in which it was stored on the computer's hard disk, to a "standard" form, representative of a well presented business letter. To complete this conversion the subjects had to perform 12 sub-tasks (see table 12.1). These were presented in the same order to each subject.

1. Getting the letter up on the screen.
2. Moving the sender's address to the centre of the page.
3. Changing the format of the sender's address.
4. Changing the font size of the sender's address.
5. Moving the date flush right on the page.
6. Putting the receiver's address into bold lettering.
7. Scrolling through the letter.
8. Deleting a word from the letter.
9. Inserting a word into the letter.
10. Moving text.
11. Saving the changes made.
12. Removing the letter from the screen.

Table 12.1. Sub-tasks performed in converting letter.

12.1.3 Procedure

After reading a short introduction to the experiment (appendix 12.1) and being shown examples of plain and standard letters (appendix 12.2), subjects were given a basic instruction session on the word processor. They were shown what the cursor was, how to position the cursor, how to highlight a word (when using Microsoft Word it is often necessary to highlight text in order to select it for alteration or manipulation), and how to pick a command from the menu. No task specific training was given.

After being issued with the instruction manuals that came with the word processor, and a description of the sub-tasks required in order to standardise the letter (appendix 12.3), subjects were advised that they could either start the first sub-task straight away, or spend some time familiarising themselves with the word processor first. Any time spent on this was included in the overall task

completion time. Subjects were told that they should attempt sub-tasks in the order in which they were described. If subjects got stuck (which they were told meant not knowing what to do for more than about two minutes), they could ask for help from the experimenter. If help was requested then the experimenter intervened to help the user complete the sub-task with which they were having difficulty. The level of intervention depended upon how far the subject had progressed before getting stuck. It ranged from advising the subject to consult the manual, if she had not already done so, through to "talking the subject through" a sub-task.

Sub-task completion times were recorded, as were any errors or interventions from the experimenter. The experimenter followed a written protocol throughout the experimental session (appendix 12.4).

12.2 RE-ANALYSING FOR THE EFFECTS OF CONSISTENCY

For the new analysis, rules for performing each of the sub-tasks were written down. These were then analysed for possible set and rule incompatibilities, and potential consistency-related usability problems were anticipated. By referring to the records from the original experiment, it was possible to see if any of these anticipated problems actually occurred.

The design of the interface meant that it was possible to perform some of the sub-tasks in more than one way. Where this was the case the rule given is derived from the procedure given in the instruction manual. The manual was available to subjects throughout the experiment. Note that the "Problems found" section only reports on those predicted from the consistency analysis, rather than every problem found. As will be seen, the consistency analysis could account for almost half of the problems encountered.

1. Opening a file
Rule: Open FILE menu + select OPEN + click on FILENAME + click on button marked OPEN.

Anticipated consistency-related problems (ACRPs): During the pre-experimental introductory session, subjects were introduced to the general principles of Macintosh interfaces, so the "pointing and clicking" nature of the task was unlikely to prove a rule incompatibility. As they had done no previous tasks with

the interface there could be no set incompatibilities. Therefore, no consistency related problems were anticipated.

2. Centring the address
Rule: Open FORMAT menu + select SHOW RULER + Highlight TEXT + click centring icon.

ACRPs: "SHOW RULER" is not an obvious name for the task of centring. There is therefore a rule incompatibility, possibly leading to users activating commands with names more suggestive of centring. There was also a possibility that users might think of centring as moving text. If they were to look up "moving text" in the manual, they would read about the commands "CUT" and "PASTE". However, these commands cannot be used for this type of "moving".

Problems found[4]: Eight of the twelve subjects either tried using the wrong command, or could not find the correct command — 4 used "POSITION" from the "FORMAT" menu, 3 used "PARAGRAPH" from the "FORMAT" menu, whilst another expected to be able to find a command called centring. Three of the subjects tried using "CUT" and "PASTE".

3. Putting heading into shadow style
Rule: (Highlight text) + Open FORMAT menu + select SHADOW

ACRPs: With sub-task 2, users could highlight the text to be moved after they had chosen the appropriate command from the menu (but before clicking on the centring icon). It is possible, therefore, that users might expect that highlighting could be done after the menu command was chosen. If this was the case then users may activate the command without first highlighting — the result of which would be that nothing would happen. An error of this type would be the result of a set incompatibility — tasks 2 and 3 being treated differently in the interface (i.e. having a different rule).

Problems found: Six subjects did not highlight the text before activating the command. In fact users needn't have re-highlighted at all, as the text should still be highlighted from the previous task (hence the brackets surrounding the first part

[4] Remember that the "problems found" sections report only those which had been predicted from the consistency analysis, rather than every problem found.

of the rule), however, many lost the original highlight in the process of attempting the task.

Indeed, having to highlight text before specifying the operation to be performed might also constitute a rule incompatibility for some subjects. This is because, here, the argument (the text to be manipulated) has to be specified before the command name. This order is contrary to that in many command based environments (and, indeed, the usual order in spoken or written English). For example, in DOS, if, say, a file were to be copied, then the user would first type the command "copy", followed by the filename (the argument), rather than the other way around. Thus, there is a possibility that subjects might bring such prior expectations to the interface, on the basis of previous computer experience.

4. Putting text into 18 point font
Rule: (Highlight text) + Open FONT menu + select 18 POINT

ACRPs: The set incompatibility with task 2, in terms of the stage at which highlighting occurs may still cause problems, although the set compatibility with task 3 on this issue, should reduce the problem. If users thought of enlarging the font as being similar to changing its style, there would be a further set incompatibility with task 3, this might lead to users opening the "FORMAT" rather then the "FONT" menu.

Problems found: Two subjects did not have the text highlighted before choosing the menu command. Again, it was not really necessary to re-highlight at all, but for many the highlight from the previous task was lost. No instances of users opening the wrong menu were recorded. However, this is possibly because this may have been regarded as a too trivial an error in the context of the original study to be worth recording.

5. Moving text to the right of the screen
Rule: (Open FORMAT menu + select SHOW RULER) + Highlight text + click right alignment icon

ACRPs: Because of the word processor's state when the user starts this part of the task the first two parts of the rule are unnecessary. This leads to a set incompatibility with the rule for task 2. Therefore, it might be expected that users would go to the "FORMAT" menu and activate the ruler command (which in this

state would be "HIDE RULER"), this would lead to the ruler disappearing. Another set incompatibility may occur if users were to think of this as a moving task, which might lead to invocation of the "CUT" and "PASTE" sequence. The way in which the task had been worded ("Move text to right of screen") might accentuate this effect. As with sub-task 2, many users may expect a menu command to be used to perform this task, rather than the ruler — another rule incompatibility.

Problems found: Five of the subjects initially hid the ruler, 4 tried using the command "Position", from the "FORMAT" menu, whilst 5 tried to use "CUT" and "PASTE".

6. Putting text into bold writing
Rule: Highlight text + Open FORMAT menu + select BOLD

ACRPs: There do not appear to be any obvious set or rule incompatibilities here. The only possibilities are that the state of the word processor (i.e. text not highlighted) might make this sub-task set incompatible with sub-tasks 4 and 5, where the prior state of the machine rendered re-highlighting unnecessary, however this has to be weighed against the set compatibility with other tasks (2 and 5) where it was necessary. The place at which highlighting appears in the rule is also set incompatible with tasks 2 and 5, but is set compatible with tasks 3 and 4. As discussed in the context of sub-task 5, the stage at which highlighting takes place might also be a rule incompatibility for some subjects.

Problems found: Three subjects did not highlight the text before activating the command.

7. Scrolling down until opening line of letter (i.e. Dear Mr / Ms .....) is at the top of the screen
Rule: Move mouse over scroll bar arrow at bottom right of screen + keep mouse button depressed until opening line at top of screen.

ACRPs: All previous sub-tasks have used either the menus or the icons on the ruler. If a subject were to regard this sub-task as being of a similar type to any performed previously, then having to use the scroll-bar, as opposed to either of the other two media, would constitute a set incompatibility. Aside from set and rule compatibility, using the scroll bar might also be considered a disadvantage in

terms of increasing the total number of different types of rules employed by an interface. As discussed in chapter 5 this may be a set independent measure of the burden associated with using an interface.

Problems found: Three subjects tried using the menus, two tried using the ruler.

8. Deleting a word from the letter
Rule: Position the cursor after the word to be deleted + hold down DELETE key until word deleted.

ACRPs: Like sub-task 7 this differs from sub-tasks 1 — 6 in that it does not rely on menus or the ruler. However, this may not be a set incompatibility, as it seems intuitively likely that users may regard deleting text as a keyboard based task, and hence as being in a different set to the others (Indeed they were informed that they might need to use the keyboard for sub-tasks 8 and 9). However, there is likely to be a rule incompatibility due to this word processor's convention of deleting the character before, as opposed to after, the cursor. Word processors vary as to whether their delete functions remove text in-front of, or behind, the cursor. Those users who were used to word processors using the opposite convention could make the mistake of positioning the cursor before, rather than after, the word to be deleted.

Problems found: Five users put the cursor before the word to be deleted. One subject tried using the menus.

9. Insert a missing word into the letter
Rule: Position cursor in appropriate place + type word.

ACRPs: There may be a rule incompatibility centring on this word processor's convention of inserting, rather than overwriting words — again conventions vary between word processing packages. This would make the task rule incompatible for those whose experience was largely with word processors of the overwriting type, who might expect to have to create space for the word, as would be necessary, for example, on a typewriter.

Problems found: Three subjects typed in unnecessary spaces.

10. Moving text

Rule: Highlight text + Open EDIT menu + select CUT + Position cursor where text to be inserted + Open EDIT menu + select PASTE

ACRPs: For those who regarded centring and right aligning (sub-tasks 3 and 5) as moving tasks there is likely to be a set incompatibility — those sub-tasks were done with the ruler, whereas this uses menu commands.

Problems found: Four of the subjects tried to perform the task using the ruler icons.

11. Saving the changes

Rule: Open FILE menu + select SAVE

ACRPs: The method for performing this sub-task differs from nearly all others in that it requires neither highlighting, nor use of the ruler, nor use of the scroll bar — one or a combination of these was required for all tasks so far, except for sub-task 1, which differed in that the name of the file to be opened had to be selected from a lower level menu. If subjects thought of this task as being grouped with any of these, then this would constitute a set incompatibility.

Problems found: None.

This result initially seems surprising in the light of the analysis (hence the comment here). After all, the action rule for this command differed from all those for previous tasks. However, it may be that subjects did not expect this to be treated as other tasks — saving is an operation performed on a file as a whole, rather than a formatting or editing task within a file. In this sense, saving is similar to opening a file. The rules for saving and opening files differ only in that the rule for opening is longer, requiring two further actions subsequent to invoking the appropriate menu command. This, however, would not be a problem in the context of saving, as even if subjects expected the rule to be longer, this would still not lead to an error — rather the task would simply be completed sooner than expected. So, contrary, perhaps, to initial impressions, the lack of problems found here may underline the importance of users' groupings. It appears that performance has been unaffected by previous tasks which subjects may have placed in orthoginal sets.

12. Removing the letter from the screen
Rule: Open FILE menu + select CLOSE

ACRPs: As this is set compatible with sub-task 11, it seems unlikely that there would be any consistency related problems.

Problems found: None.

In total the experimenter had recorded 148 errors made by the twelve subjects over the twelve sub-tasks. Sixty-six (45%) of these were of a type predictable from the consistency analysis. Of the 48 which were serious enough to require experimenter intervention 23 (48%) could have been predicted from the above analysis. Thus, almost half of the problems identified may be attributable to set or rule incompatibilities. Unfortunately time penalties induced by each individual problem were not logged in the experimental notes. It was not possible, therefore, to quantify the detrimental effects of inconsistencies in terms of time.

12.3 DISCUSSION

From this re-analysis it appears that the answers to the research questions are:

Yes, designers do really build inconsistent interfaces, and

Yes, this does matter in a "real-life" context, certainly, at least, for first time users.

So, why would a designer build an inconsistent interface? Perhaps the answer is that the inconsistencies may not be intuitively obvious. Indeed this author's original write up of the study (Jordan 1992) makes no mention of inconsistencies as a possible cause of user difficulty. Superficially, the interface appears consistent — for example, the mouse can always be used to highlight text and select commands from menus, all the alignment commands can be found on the ruler. It is only by breaking tasks down into action-rules that some of the more subtle inconsistencies are exposed.

It could also be that this is inherently a guessability type problem, which might not be discovered by one-point usability evaluations. Although many of the problems

found were due to set incompatibilities, these might not necessarily be persistent in an interface, as users might restructure their sets in the light of experience. Evidence from the previously reported laboratory studies suggests that effects from set incompatibilities nearly always persist beyond guessability. However, the tasks chosen for those experiments were ones where particularly strong associations were expected. As possible future work, a longitudinal study might follow users performing with a real interface over a period of time to see if these types of problems persist.

Even if these problems did only affect guessability, they would still not be trivial. An un-guessable interface can be off-putting to users, and indeed the Macintosh operating environment appears to owe much of its popularity to its perceived accessibility to the new user (Joseph 1991). Further, the word processor studied here can be used for, at a conservative estimate, at least two hundred tasks. Clearly, if a user is to make use of this range of functionality, he or she will have to go through the guessability stage with each. Thus, even in the long term of interface usage, guessability problems could be serious.

The percentage of problems in this interface which could be attributed to inconsistencies seems surprisingly high. Whilst it might be expected that inconsistencies would impede performance, there are many other potential design faults which might also be expected to have an effect (Ravden and Johnson (1989), for example, discuss some of these: visual clarity, feedback, explicitness, flexibility and control, error prevention, user guidance). It may be that some of the problems, here attributed to inconsistencies, are partly due to other, compounding, design faults. For example, it might be argued that difficulties in centring text were caused partly by the invisibility of the appropriate icon whilst the ruler was closed. However, another possible reason for the large effect of consistency issues, might be that most other potential sources of difficulty have been eliminated by the designers. For example, in comparison to many other types of interface, the Macintosh interface gives excellent feedback, clear prompts and error messages, and has a well laid out screen with the menu headings clearly displayed.

Even if the strength of the effects found from the re-analysis for consistency could not be generalised to other types of interface, the results indicate that inconsistencies can have large effects on the performance of realistic tasks using a widely used interface. The analysis has brought out what appear to be fairly arbitrary design decisions — with some formatting being done by rulers and some

by menus. It has also shown that a lot of the problems concern an issue not well addressed in the experiments reported earlier in this thesis, or indeed most other consistency work (the exception, perhaps, being work with Production Rules (Kieras and Polson 1985)): conditional task-action rules (e.g.: if text is not highlighted then highlight, if ruler not visible then make it visible).

12.4 SUMMARY

A study was designed to address the questions of whether designers really designed inconsistent interfaces, and if so whether this had a marked effect on usability. By breaking down the methods for performing tasks into action-rules, and by analysing these for possible set and rule incompatibilities, almost half of the problems found, associated with the guessability of a commercially available interface, were predicted. The sub-tasks set were realistic, and the subjects representative of the target user group. Thus, the results indicate that the effects of set and rule incompatibilities may be generalisable from the tightly controlled conditions artificial laboratory experiments, to more realistic contexts.

Chapter 13

CONCLUSIONS

13.1 GENERAL OUTCOMES

A multi-component usability framework was proposed, the various components describing how a user's performance on a task changes with experience. An experimental methodology and criteria based analysis method for measuring the various components of usability was outlined and demonstrated. This was then applied in a series of studies investigating the effects of two distinct types of consistency on the various components of usability.

The methodology involved taking performance measures over a series of trials, making it costly in terms of investigator time and effort as compared to traditional one-point studies. However, there is no obvious experimental alternative for quantifying the various usability components, as with one-point studies it may not be clear which is being measured. The criteria based analysis method was fairly quick and easy to apply. It was less cumbersome to implement than most of the alternatives considered, and avoided some of the theoretical difficulties associated with others. Prospects seem good for applying non-experimental methodologies with the usability framework — this is discussed further in the "Future research" section of this chapter.

A series of predictions were made about how the various components of usability would be affected by set and rule compatibility. These were based on the expectation that rule compatibility would have a greater influence during users' early interactions, whilst set compatibility would be more salient later on. Generally, the outcomes of the studies provided strong support for this, across different types of interface, different levels of an interface, and different levels of task. There were, however, some failures of prediction. A proportion of these may have been attributable to small sample sizes, which meant that possible effects may not have been confirmed by statistical tests, whilst some may have been due to experimental noise which could have swamped effects. However, others could not be explained away like this, and appeared to be real failures of prediction. In particular, little evidence of effects for rule compatibility on learnability was found in situations where set compatibility was absent.

A criticism of the first four consistency studies is that they lack ecological validity. The interfaces used were artificially created for the experiments, subjects were set very artificial tasks — simply invoking commands, and it was doubtful that subjects would have had much motivation beyond, perhaps, a desire to complete the experiment as quickly as possible. Certainly, they would not have learned any particularly useful skills. These criticisms also apply to much of the rest of the consistency literature. However, in the studies reported here it was further assumed that users would form the associations which influence consistency simply on the basis of command names, menu headings, or icon labels (depending on the study), rather than on the functions of commands, which were, in fact, not implemented. Indeed, the feedback given by the experimental interfaces gave little indication of what the consequences of command invocation would be in more realistic situations. Here, then, a further type of artificiality had been introduced. This left the work undefended against a common criticism of HMI research — that effects demonstrated in artificial studies might not be important in realistic contexts.

The last of the reported studies, and, to a degree the penultimate one, were motivated by the need to address this issue. In the final study data from representative users of a commercially available interface was re-analysed, indicating large effects for set and rule incompatibility. Indeed, these were apparently associated with almost half of users' problems. These subjects were performing comparatively complex tasks reflecting an aspect of their daily working practice. The penultimate study was more tightly controlled, but, again, realistic tasks were set, subjects were given clear instructions as to the effects they were to achieve, and the interface — which was only slightly altered from its original form — gave clear feedback. Again, results from this study strongly supported the predictions it was designed to test. The outcomes of these two studies, then, indicate that it may not be necessary to be over-cautious about generalising from the results of the earlier ones.

However a qualification on the effects of consistency is that they appear to be dependent on users' meta-level strategy. In all of the studies reported individual subjects could be identified whose performance deviated from what was predicted, even when the sample as a whole performed as expected. It was suggested that this might have been the result of these users' strategies. When the predictions about the effects of consistency were made, they were based on assumptions that

users would put tasks into sets, and that they would have a priori expectations about the action-rules required. Clearly, if these assumptions are not valid for a particular user, then set and rule compatibility would not be expected to have any effect. In fact, the concepts would be meaningless in relation to such a user. After all, if a user has no expectations about schema sets there is nothing to compare with designers' sets. Similarly, without a priori expectations about action-rules, rule compatibility can not be an issue.

These individual differences reinforce the idea that consistency must be considered in relation to both the interface and the user. In turn, this supports the idea that the set compatibility / rule compatibility distinction is the right one, as opposed to distinctions such as internal / external consistency. The latter distinction implies that inconsistencies can be identified in isolation from the user, either from mismatches within the interface, or mismatches between the interface and some aspect of the outside world.

Another implication of these individual differences is that, strictly, usability can only be considered in terms of each individual user. Equally, however, the overall support for the experimental predictions implies that, at least in the context of set and rule compatibility issues, it is possible to make meaningful generalisations across groups of users as a whole, provided that groups are defined by experience (i.e. that the component of usability is specified). However, these experiments were designed with the intention that task-task and task-action associations should be obvious and predictable, thus inter-user differences may have been less than might have been expected in other situations. Similarly, the extent to which an interface's usability varies between tasks is likely to be dependent on the tasks involved. For example, where the range of tasks supported by an interface require similar action-rules, users' performance on one is likely to reflect performance on another, and experience on the interface as a whole may be transferable between tasks. This was illustrated by the results of the experiments investigating the predictions about the effects of consistency on guessability with and without previous experience of the interface. Conversely, where different tasks required different types of action-rules, experience may not be transferable and performance may vary considerably between tasks.

Whereas the majority of previous work on consistency related issues was done in the context of recall based interfaces, that reported here was mainly in the context of recognition based ones. An issue, then, is the question of differences between

recognition and recall based interfaces. A point originally raised by Furnas et al. (1987), and discussed in the context of the recall based experiment reported, was that command names on which a high percentage of the user population agree can rarely be found. This implies that recall based interfaces will be inherently unguessable, as it will be difficult to design them such that tasks are rule compatible. This problem was avoided in the reported study, as instructions were given in such a way as to suggest particular command names. Even then, however, two of the eight subjects consulted help before completing (what it was anticipated would be) rule compatible tasks for the first time. Where command names which bore no resemblance to their associated function were used, no subject was able to complete the experimental tasks for the first time without consulting help. By contrast, with the recognition based interfaces, users were able to succeed at tasks, even if the rule was not immediately obvious, by exploration.

It may be that the difference between recall and recognition based interfaces is the comparative size of problem space. With recognition based interfaces the user has to choose a menu heading / menu item / icon from those displayed on the screen. With recall based interfaces however, the problem space is not in the interface, but in the users head — it may possibly include the user's entire mental contents. Obviously, this is vast by comparison. So if users of a command based interface were trying to infer or recall a command for, say, deleting there might be a vast array of competing alternatives in their minds, for example: "erase", "delete", "remove", "cut", "clear", "rub out" or "efface". Even then, it may turn out that what is required is not an English word but rather an abbreviation, or command-key sequence. With recognition based interfaces however, it may be fairly obvious which of the items available is appropriate, as perhaps only one or two could plausibly be associated with the task in question.

On the basis of work done with recall based interfaces, Landauer (1987) was pessimistic about the prospects for effects found from laboratory based experimentation being generalisable to more realistic situations. For example, he had found significant advantages for some command names over others, but found these effects were not big enough to be important outside the laboratory. A possible reason is that, as Furnas et al. (1987) demonstrated, comparative advantages for some commands over others may only be slight, and are subject to a high degree of inter-user variation. With recognition based interfaces, however, where users are presented with a limited choice of possible command names and potential action-rules, what is expected for a particular task might be comparatively

robust, and less variable between users. Thus, it might be reasonable to be more optimistic about the chances of effects demonstrated in artificial contexts surviving in more realistic situations. Indeed, this is supported by the outcome of the last of the reported studies.

13.1.1 Implications of the usability framework for design and evaluation

Applying the multi-component usability framework to investigate the effects of consistency gave insights that measurement at a single point during a user's learning could not. Where traditional usability measurement techniques give a "snapshot" of user performance, the proposed methodology gives a profile. This should enable investigators not only to identify properties which affect usability, but also to diagnose the nature of their effects. A consequence of this should be to facilitate more focussed use of design resources. For example, from the work reported in this thesis, it would appear that when designing a product for one-off or occasional use, say a tourist information system, the designer should pay particular attention to rule compatibility. Conversely, if EUP is the important factor, say with a CNC milling machine, where the operator will receive training, then designing for set compatibility should be a priority.

A counter-argument to the above might be that a good designer should adhere to all the principles for usable design, and so giving priority is not an issue. However such an argument seems unrealistic on two counts. Firstly, usability issues may be competing for the designer's attention with many others — for example, aesthetics, functionality, manufacturing and design costs. If the designer has only a limited time to dedicate to usability issues, it is not reasonable to expect him or her to consider everything. Sets of guidelines, listing the requirements for usable designs can often appear to be excessively long — the 163 items listed by Marshall, Nelson, and Gardner (1987) not being untypical. As Booth (1989) pointed out, it seems unrealistic to expect such exhaustive lists to have much influence on designers. However, if the usability framework were used, and guidelines were made more component specific, it seems more likely that they might receive attention.

Secondly, as was suggested in chapter 2, situations may arise where adherence to one principle might lead to the violation of another, requiring the designer to

consider trade-offs. An example given was that whilst displaying a high proportion of an interface's functionality on a screen might be good for guessability, subsequent visual clutter might not be good for EUP. Employing the multi-component framework can help classify the nature of any potential trade-offs to be made (e.g. learnability at the expense of system potential, or EUP at the expense of guessability), and help in the making of design decisions.

From the evaluator's point of view the multi-component framework casts doubts on the validity of traditional one-point evaluations, where it can often be un-clear as to which component is being measured. Application of the multi-component framework can help the evaluator build a comprehensive picture of performance with an interface over time, making statements about an interface's usability, interface comparisons, and classification of the effects of bugs more meaningful.

Consider, for example, the reported study on the VCR. Imagine that a one-point evaluation were to be done with this, with subjects being given a practice session beforehand. Suppose that this practice was sufficient that they reached a level of performance equivalent to that on, say, trial four of the experiment as it was run in the study reported. By comparison with the reported study, this one-point evaluation would be an impoverished source of information. For example, nothing would have been discovered about the special difficulties associated with the first attempts at new tasks. It would be difficult to know if any bugs discovered would have longer term effects, and many of the bugs which affected performance during earliest interactions would have been missed. Imagine that this VCR were to be compared to another, and that this had undergone a similar one-point evaluation. The results might show an advantage for one of the two VCRs, but what would this mean? It might be, for example, that users of one of the two were not yet passed the guessability stage whilst the others were half way to EUP, or that one group were at EUP whilst the other had just passed the guessability stage. Either way, it is difficult to know what measurements taken on a one-off basis really stand for and is likely to be misleading if taken as a guide to the respective usabilities of different interfaces.

Initially, the advantages gained through application of the multi-component framework might appear to be tempered by concern about the cost involved in applying the appropriate methodology. After all, building up a performance profile requires task repetition, and the experiments reported here involved a minimum of ten trials being performed so that guessability, learnability, and EUP could be

measured. Of course, building a performance curve will take several trials, however there may often be situations in which the evaluator is only interested in a specific component of usability. In such circumstances evaluations could be less costly. For example, if guessability were the only component of interest, subjects would merely have to perform the experimental tasks once, whilst if EUP were being investigated subjects with a suitable level of experience of the tasks could be recruited, and again one trial may be enough. Reference to the framework is still important here, as the design of the experiment, and perhaps the characteristics of the subjects required may be dependent on which component is of interest. In particular, pre-experimental practice needs can be addressed. A common approach in current HCI evaluation is to give subjects apparently arbitrary amounts of free practice time — as was suggested in the imaginary one-point evaluation of the VCR discussed above. By considering the component of usability to be measured, the purpose, and therefore the length and content, of the practice session should become clearer.

It should also be noted that the experiment is not the only means of usability measurement. In chapter 3 a variety of empirical, and non-empirical alternatives were listed, many of which are less costly in terms of investigator time and effort. Jordan and Kerr (1993) evaluated a multi-function telephone system with respect to each component of usability without performing an experiment, but rather using a feature checklist, think-aloud protocols, and a questionnaire. With instruments such as these, evaluating several or all of the components need not be much more costly to the evaluator then measuring just one. In the case of, say, a questionnaire or semi-structured interview, this would merely be a matter of asking additional questions. Adapting non-experimental methods for use with the multi-component framework is discussed later as a possible goal of future research.

13.1.2 Implications of set and rule compatibility

Two distinct types of consistency have been identified and defined. Experimental evidence suggests that each affects usability in different ways, and that these effects appear to be generalisable to different types of interface, different levels of an interface, and different levels of task.

Although set and rule compatibility have never been explicitly discussed before, they have been derived from an increased understanding of consistency issues

which has evolved through successive treatments, and the idea of two types of consistency is not a new one. For example, Kellogg (1987) refers to internal and external consistency — a distinction rejected here — whilst Payne and Green (1986) recognised that the number of rules needed for a set of tasks is a separate issue from their a priori conformity to users' expectations. What has not been recognised before, however, is that different types of consistency can affect usability in different ways.

Payne and Green's (1986) use of the "known-item" notation in TAG, for example, implies that a priori rule predictability should be used as a secondary indicator of usability, behind their primary measure — the number of rules. So when comparing two interfaces they would suggest the one requiring the fewer action-rules was more usable, but that if the interfaces were equal in this respect then a priori predictability could act as a "tie-breaker". Implicitly, this suggests that, whilst set compatibility is important, rule compatibility is a mere nicety by comparison. Whilst this may be the case when considering EUP, it certainly isn't during earlier stages of interaction. Indeed, during the earliest, rule compatibility is more important. So, whilst set and rule compatibility are related issues, in that both are concerned with the action-rules associated with tasks, they are, in terms of their effects on usability, independent. Indeed, the effects of rule compatibility on performance may be more closely related to quite different properties of an interface — say, explicitness, or feedback — than to set compatibility. The mistake made by previous work, it appears, was to assume the effects of different types of consistency to be similar, as the issues derive from a similar root. It was only through employing the multi-component usability model, that the difference in effects was highlighted.

The concepts of set and rule compatibility embody the important point made by Reisner (1990), that consistency is concerned with matches, or mismatches between a user's mental contents and properties of a design, rather than being a property of a design alone. Although this point was discussed in chapter 5, the design of subsequent studies relied on the investigator being able to predict users' task groupings and rule expectations. It was assumed that by using task sets which apparently had strong natural groupings and rule associations, users could be expected to behave in a fairly uniform and predictable manner. For example, in the first consistency study, it was assumed that pairings such as "invoke "New Window"" with "invoke "Zoom Window"" would be universal, and that users would expect to find these commands on the menu titled "WINDOW".

Results indicated that, generally, these assumptions were fairly reasonable. However, in each of the experiments designed for within subjects comparison, there were always some subjects whose comparative performance on the different experimental tasks was not as expected. The most noticeable instance of this was in the first consistency experiment, where, when taken as a whole, male subjects behaviour at EUP seemed influenced by rule but not set compatibility — exactly the opposite of expectations. The reason suggested for this was that perhaps these subjects were using a different strategy to the others. Because the interface as a whole contained many set incompatibilities, they may have decided that they would be better off trying to learn the rules for tasks individually, rather than trying to remember any common schemata. However, whatever the reason, these subjects did not appear to be using task groupings in the predicted manner.

This reinforces the idea that some sort of discovery procedure may be necessary, in order to form clear ideas about users' task groupings and rule expectations. After all, if these could not be always be predicted in apparently clear cut cases, it seems even less likely that they could be where there might be an element of ambiguity. This is discussed further in the next section on possibilities for future research.

13.2 FUTURE RESEARCH

During the course of the research reported in this thesis some issues arose which were not tackled in depth, as, although they might be important, they were not regarded as being central to the theme of this thesis.

13.2.1 Developing non-experimental instruments for usability measurement

Perhaps the least appealing aspect of the multi-component framework, is the costly experimental methodology associated with it. As previously discussed, this cost can be significantly reduced if the investigator decides to focus on a particular component of usability, rather than measuring the whole performance profile. However, non-experimental techniques can be another way of reducing the cost to the investigator. The most commonly used of these were outlined in chapter 3. So

how could these be used to measure the various usability components. Some of the empirical methods rely on asking users questions — questionnaires, focus groups, semi-structured interviews, incident diaries (Jordan (1992b) calls these survey analyses). By asking the appropriate questions it might be possible to obtain measures relating to particular usability components. However, deciding what these questions are is a matter for research, as presumably responses would have to be validated against behavioural measures to see if they were really reflecting, say, performance or satisfaction (remember performance is not the only measure of usability). Through discovering which design issues affect each component, it might also be possible to make accurate forecasts about an interface's usability via predictive methods, such as task analyses, or expert walkthroughs.

13.2.2 Total experience with an interface vs. individual task experience

One of the predictions about the effects of consistency on usability was that set compatibility would be an important determinant of guessability, for those with previous experience of using the interface for similar tasks. Experimental results supported this. The effects for set compatibility here mirror those at EUP, where the user has repeated a particular task several times. It appears, then, that in this context, performance on an individual task can be influenced by previous experience of other similar tasks on the interface, in the same way as it would be by experience of the particular task itself. Whilst this effect may be peculiar to the properties of set compatibility, it could be that, in many cases, the issue is fundamental to the nature of experience in HCI.

For set compatibility, then it seems that total experience of an interface with a task set may be the most important issue. A research question is whether this is the case in relation to other design properties, or whether it is peculiar to set compatibility. With rule compatibility, for example, effects on guessability for experienced users did not mirror those at EUP in this way.

13.2.3 Developing a discovery procedure to identify inconsistencies

Fundamental to the definition of set and rule compatibility is the idea that consistency is to do with matches or mismatches between the user's prior mental contents and a design, rather than merely being a property of the design alone. Whilst design properties might be investigated via, for example, an expert walkthrough, this may not be successful at identifying inconsistencies, without knowledge of users' mental contents against which to compare.

A discovery procedure should capture users' settings and rule assignments, so that these could then be used in formalisms — taking the place of the investigator's assumptions about users' anticipated schema sets and rule schemata, which were used in the formalisms describing the experimental interfaces for the studies reported in this thesis. If an efficient discovery procedure were to be developed, this might offer significant advantages over having to use behavioural measures to identify inconsistencies. However, this is still removed from the overly optimistic approach taken by most formal grammars, where the investigator's judgement as to users' settings and expectations is relied upon.

13.2.4 Strength of affinity

Set and rule compatibility are, respectively, connected with affinities between different tasks, and affinities between tasks and action-rules. In the experiments reported here, set and rule compatibility have been manipulated depending upon whether the direction of users' expected affinities are the same as those embodied in the interface. What has not been investigated, however, are possible effects due to the strengths of affinities.

Experimental tasks were chosen here with the intention that settings and rule expectations would be fairly unambiguous. By definition, this implies that affinities would be fairly strong. A question, then, is what might happen if affinities were weaker. It might be expected, for example, that the benefits of set and rule compatibility may not be so great, or that, by corollary, inconsistencies may cause less severe problems. However, this may not necessarily be so. For example, for any individual combination of user and interface, it may be that set and rule compatibility are binary issues — the interface is either consistent or it

isn't. This may be similar, say, to asking whether feedback messages on an interface are readable. As long as character size is large enough to read, messages may be useful. However, further increases beyond this may not necessarily bring greater benefits. Experimental manipulation of strength of affinity might help address these issues. Again, a discovery procedure would be necessary for a priori investigation of affinity strength.

13.2.5 Percentage set compatibility

An issue connected with set compatibility, is what percentage of tasks in a user's set are treated the same way by an interface. Consider, for example, the task of invoking the character styling command "BOLD". If this were contained on, say, a menu headed 'FONT", whilst the other styling commands "ITALIC", "OUTLINE", and "SHADOW" were contained on the "FORMAT" menu, users would find it incompatible with the others in its set, and might be expected to have problems with it at EUP. Now imagine that, say, "OUTLINE" were also to be moved to the "FONT" menu. Would this make it easier or more difficult for users to remember how to invoke "BOLD"?

There appear to be two intuitively reasonable ways of making a prediction. The first is to suggest that it will be easier to remember as the schema "Open FONT menu + select X" now accounts for half the tasks in the set and is therefore worth remembering, even if there may be problems remembering which tasks this operates on. It may be that if only "BOLD" could be invoked using this schema it would a) be rarely used, and thus under-rehearsed, and b) not be worth learning anyway, as it is useful for only a single task. A second approach, though, might be to suggest it would be more difficult to invoke, as the problem of deciding which schema should be used for which tasks has become more complex. Before, the user would only have had to remember that character styling commands were on the "FORMAT" menu, with the exception of "BOLD" on the "FONT" menu. Now, however, they may have to remember command menu placements on an individual basis.

13.2.6 Investigating user strategy

It was suggested that, in the first consistency study, the reason for differences in performance between two groups of users may have been due to differences in learning strategy. It was implied that some of the subjects may have considered tasks individually, rather than in groups, and moreover, that this was a deliberate strategy, chosen to counter set incompatibilities.

It might be interesting to test these assumptions empirically. For example, it might be that the comparative effects of set and rule compatibility could be reversed simply by asking users to adopt a particular strategy. If this were true potential problems might be overcome, say, through instructions in the manual, or through training. Conversely, strategy may be a matter connected with properties of the individual — indeed the identification of two groups by gender in the first consistency study seems to suggest this. This might mean that either users couldn't deviate from their natural strategies on request, or would suffer a decrement in performance if they attempted to do so. This would be similar, say, to asking a right-handed person to operate the mouse left-handed.

13.2.7 Manipulation of prior mental contents

If consistency is to do with the interaction between a user's mental contents and a design, then it might be possible to manipulate consistency, not only through interface design, but also by manipulating mental contents. Indeed, this strategy was used in the study reported in chapter 11, where subjects were given practice on a statistics package in an attempt to influence ideas about rule compatibility when they came to use a word processor. This was an example of attempting to manipulate prior mental contents through experience.

It might also be possible to influence prior mental contents via an interface's related documentation, perhaps by describing a particular model of how the interface operates. For example, simply describing an interface as menu driven is likely to make command activation from the menus more rule compatible than using, say, icons. Similarly, it may be possible to influence task settings. Consider, for example, a word processor. If users were told that this had two main types of functionality — "formatting" and "editing" — then they might

expect, say, putting text into bold style writing to be similar to "formatting" tasks, such as margin setting, rather than editing tasks, such as deleting text. However, if the user were told that the two types of task were, say, "text operations" and "document layout", then they might expect putting text into bold type to be similar to deleting, rather than margin setting, as the first two are operations done on text strings. So, if prior mental contents could be manipulated in this way, it might be possible to remove some inconsistencies without changes to the interface.

Another way of influencing user expectations might be through employing metaphors. It may be, for example, that users who come to the Macintosh with the idea that they should treat the screen as if it were a desk top, will find this helpful in some ways, but a hindrance in others. This metaphor might make file deletion rule compatible — to do this users drag a file icon to a bin icon. This mirrors something which can be done in the context of using a desk — removing something unwanted from it, and dropping it in a bin. However, the metaphor may be of little use when a menu item is needed. After all there appears to be no obvious parallel between this and desk use.

Appendix 4.1

Overview of the Experiment

The purpose of this study is to evaluate the usability of a video cassette recorder (VCR). As a subject in this experiment you will be asked to do some tasks using this video recorder. Your performance will be looked at so that a rating can be gained for the usability of the VCR.

The experiment will take about an hour and a half in total, split into two sessions — todays session will be by far the longer of the two. In each session you will be asked to complete a series of tasks with the VCR whilst the experimenter observes and takes notes.

The tasks that you will be given have been chosen with the intention that they are representative of how someone would typically use a video recorder — these will be explained to you in more detail when we are ready to start. If you get stuck during the experiment *(being stuck = you do not know what to do for more than about two minutes, or feel like asking someone else what to do)*. I shall be able to give you a limited amount of help on request. **Remember it is the video recorder that is being tested not you, so don't worry if you have any difficulties or get things wrong.**

Before the experiment begins you will be given a personal details interview, which will ask for details relevant to the experiment.

The results from your particular sessions, and the details you give in the interview will be treated in the strictest confidence.

Do you have any questions?

Appendix 4.2

Example of a Task Sheet

1. Today is Wednesday the time is four o'clock in the afternoon please set the clock accordingly.

2. Put the pre-recorded cassette in the video recorder, and re-wind it to the beginning.

3. Play the video of "Skin Deep", by the Stranglers. The position of this video, on the cassette can be found on the sheet inside the cassette box, along with its counter index.

4. Stop the tape, and change the TV channel to number 4.

5. Remove the pre-recorded tape and put the blank tape into the machine.

6. Set the timer to record a program on programme position I. The programme will be transmitted on channel 3 tomorrow (Thursday) at 8.25 pm. The programme lasts one hour and ten minutes.

Appendix 4.3

Experimental Protocol

SESSION ONE

1. Give subjects "**Overview of experiment**" to read.

2. Personal details interview. Read this out to the subjects and fill in myself.

3. Introduction to session.
Say: "In a moment I am going to allow you to get started with the video recorder. I shall give you the first task sheet straight away, and you shall be given subsequent task sheets when you have completed it. You may feel that you wish to start the first task sheet straight away, or you may prefer to spend some time familiarising yourself with the video recorder first — it is entirely up to you."

4. Give the subjects the first task sheet.
Say: "This is the first of a series of ten task sheets. Please feel free to spend a few minutes reading through the sheet, before you start. If you are not clear as to what you have to do feel free to ask. You will be allowed to keep this sheet with you when doing the tasks. Please don't start doing anything with the video recorder yet"

Do not let subjects look at or touch control panel, or read manual yet.
Answer questions about what subjects have to do, but not on how they should do it.

5. Issue subjects with the manual.
Say: "This is the manual which comes with the video recorder. You may consult this as often as you wish during the experiment."

6. Give subjects advice about asking for help

Say: "If you get stuck during the experiment you may ask me for help. Getting stuck means either not knowing what to do for longer than about two minutes, or feeling as if you need to ask what to do. If you feel as if you need to ask for help then I can intervene, following a pre-set procedure. The type of help provided will depend on how far you have already progressed on your own."

7. Tell subjects that they may now start the first task sheet.

Say: "In a moment you may get started with the video recorder. Remember that you may practice with the video recorder for a while if you wish, but equally feel free to start the first task sheet straight away. I shall be sitting here making notes about aspects of the way the video recorder works, and timing how long each of the tasks takes, but remember this is a test of the video recorder not you, so try to ignore my presence as best you can — unless you want to ask a question or are stuck."

Fill in free session sheet

Fill in the task score sheet (timing every sub-task and noting any problems or errors).

If help is required then follow the help procedure and record the level of help given.

After all the task sheets have been completed or they have spent 90 minutes on the tasks stop the subjects.

8. Conclude session.

Say: "That concludes the first session of the experiment, thank you very much for your co-operation."

Remind subjects that they should return for another session tomorrow.

SESSION TWO

1. Tell subjects that they may now start to work through the task sheet.

Say: "In this session you may get started with the task straight away. Remember you may look at the manuals whenever you wish."

Issue manual.

Fill in the task score sheet (timing each sub-task and noting any problems or errors).

If help is required then follow the help procedure and record the level of help given.

2. Conclude the experiment.

Say: "That concludes the experiment, thank you very much for your co-operation."

Appendix 4.4

Action-rules for Performing Sub-tasks set with VCR

SETTING THE CLOCK TO A SPECIFIED DAY AND TIME.

1. Press the "CLOCK" button (the current day setting flashes).

2. Press the "+" or "-" button and advance or reverse through the days of the week until the required day is reached. Press the "SET" button (the current hour setting flashes).

3. Press the "+" or "-" button and advance or reverse through the hours until the required hour is reached. Press the "SET" button (the current minute setting flashes).

4. Press the "+" or "-" button and advance or reverse through the minutes until the required time is reached. Press the "SET" button (an audible beep indicates that the clock is now set).

INSERTING A PRE-RECORDED TAPE AND RE-WINDING IT TO THE BEGINNING.

1. Put the tape into the cassette compartment (tape starts to play automatically).

2. Press the "STOP" button (tape stops playing).

3. Press the "REW" button (tape re-winds to the beginning and stops automatically).

FAST FORWARD THE VIDEO TO A SPECIFIED LOCATION AND PLAY.

1. Press the "DISPLAY" button (tape's counter position is displayed — currently this is 00:00).

2. Press "FF" button (tape starts winding forward, and displayed counter position changes accordingly).

3. When the desired counter position is reached press "STOP" button (tape stops winding forward).

4. Press "PLAY" button (tape starts playing).

STOP THE TAPE AND CHANGE THE TV CHANNEL.

1. Press "STOP" button (tape stops playing).

2. Press the "DISPLAY" button (the current channel setting is displayed).

3. Press "CHANNEL ^" or "CHANNEL ˇ" button and advance or reverse through the channels until required number channel is displayed (the channel has now been changed).

EJECT THE PRE-RECORDED TAPE AND REPLACE IT WITH A BLANK ONE.

1. Press the "EJECT" button (pre-recorded tape is ejected from the machine).

2. Remove the pre-recorded tape from the cassette compartment.

3. Put the blank tape into the cassette compartment (note that the tape will not be played automatically, as the "automatic play" facility is only invoked by the insertion of pre-recorded tapes).

PROGRAM THE TIMER TO RECORD FROM A SPECIFIED CHANNEL AT A SPECIFIED TIME.

1. Press the "PROG" button (program position number starts flashing on the display — position I will be displayed). Press the "SET" button[5] (the currently set channel number flashes on the display).

2. Press the "+" or "-" buttons and advance or reverse through the channels until the required channel number is reached. Press the "SET" button (the current day setting flashes).

3. Press the "+" or "-" button and advance and reverse through the days of the week until the day when the program is to be transmitted is reached. Press the "SET" button (the current hour setting flashes).

4. Press the "+" or "-" button and advance or reverse through the hours until the hour when the program is to be transmitted is reached. Press the "SET" button (the current minute setting flashes).

5. Press the "+" or "-" button and advance or reverse through the minutes until the time when the program is to be transmitted is reached. Press the "SET" button (the default program length setting in hours starts flashing).

6. Press the "+" or "-" button and advance or reverse through the recording length in hours until the appropriate program length is reached. Press "SET" button (the default recording length setting in minutes starts flashing).

7. Press the "+" or "-" button and advance or reverse through the recording length in minutes until the appropriate length is reached. Press "SET" button (an audible beep indicates that the timer is now programmed).

---

[5]Note that in this experiment only the first program position was used.

Appendix 7.1

Balancing for Experimental Task Presentation Order, and for Task by Task Type

The order in which tasks were presented was balanced both between subjects and between experimental trials using Latin Square designs. Tasks were also balanced for task type between the four experimental interface configurations.

Overall 36 subjects participated in the experiment — these were divided into four separate groups, depending on their gender and whether or not they had used a word processor before: 16 were female with no previous word processor experience, 8 were female with previous word processor experience, 8 were male without experience, and 4 were male with experience.

Each subject was allocated one of sixteen classification codes according to the interface configuration they were to use (configuration P, configuration Q, configuration R, and configuration S), and the order in which tasks were to be performed in their experimental trials (order 1, order 2, order 3, order 4). The possible codes were, then: P1, P2, P3, P4, Q1, Q2, Q3, Q4, R1, R2, R3, R4, S1, S2, S3, and S4.

A female subject without word processing experience was allocated to each of these codes (allocation was at random). Two female subjects with with word processing experience were allocated to use each of the interfaces. This was done such that each order would also be covered twice (it was randomised within these constraints). Inexperienced male subjects were allocated to the same codes as inexperienced female subjects (at random within that constraint), whilst one male subject was allocated to each interface with one covering each of the orders (again allocation was random within these constraints). Allocation of subjects to classification codes is summarised in the table below.

Classification code	Inexperienced female subjects (n = 16)	Experienced female subjects (n = 8)	Inexperienced male subjects (n = 8)	Experienced male subjects (n = 4)
P1	1	1	1	1
P2	1	0	0	0
P3	1	1	1	0
P4	1	0	0	0
Q1	1	1	1	0
Q2	1	0	0	0
Q3	1	1	1	1
Q4	1	0	0	0
R1	1	0	0	0
R2	1	1	1	0
R3	1	0	0	0
R4	1	1	1	1
S1	1	0	0	0
S2	1	1	1	1
S3	1	0	0	0
S4	1	1	1	0

Number of subjects allocated to each classification codes.

Four different task presentation orders were used, as follows (remember, type A, type B, etc. refers to the set and rule compatibility of the task). These orders are described in the table below:

Order	1st	2nd	3rd	4th
Order W	Type A	Type B	Type D	Type C
Order X	Type B	Type C	Type A	Type D
Order Y	Type C	Type D	Type B	Type A
Order Z	Type D	Type A	Type C	Type B

Task presentation orders.

These different orders represent a balanced Latin square design. This balances for order effects by ensuring that each type of task follows every other type of task exactly as often as it precedes it. So, for example, type A tasks precede type B tasks twice (order W, and order Z), and follow them twice (order X and order Y).

To balance experimental task presentation order between subjects, the order in which the tasks were presented in each trial was varied according to classification number. To balance within subjects between trials, order of presentation was also varied according to experimental trail number. Again, both these balances were via Latin square designs. The table below describes the order in which subjects were presented with the experimental tasks in each of their experimental trials:

Trial number	**Subjects P1, Q1, R1, S 1**	**Subjects P2, Q2, R2, S 2**	**Subjects P3, Q3, R3, S 3**	**Subjects P4, Q4, R4, S 4**
1	W	X	Y	Z
2	X	Y	Z	W
3	Z	W	X	Y
4	Y	Z	W	X
5	W	X	Y	Z
6	X	Y	Z	W
7	Z	W	X	Y
8	Y	Z	W	X
9	W	X	Y	Z
10	X	Y	Z	W
11	Z	W	X	Y
12	Y	Z	W	X

Experimental task presentation order by subject by trial.

Balancing of task by task type was also via Latin square design between interface configurations as follows:

Experimental task type	Configuration P	Configuration Q	Configuration R	Configuration S
Type A	Smaller font size	Move to start of document	New window	Copy formats
Type B	Copy formats	New window	Move to start of document	Smaller font size
Type C	New window	Smaller font size	Copy formats	Move to start of document
Type D	Move to start of document	Copy formats	Smaller font size	New window

Task by task type for each experimental interface.

This balance was done in order to control for possible effects due to, for example, the position of the menu on which commands were contained, or the possibility of certain commands being more memorable than others. So, for example, it might be that subjects could remember the contents of the "WINDOW" menu more easily than, say, the contents of the "DOCUMENT" menu (as the "WINDOW" menu was at the extreme right of the screen), or that the whereabouts of "Copy formats" might prove more memorable than the position of "Smaller font size". Had these inter-interface balances not been made, this might have meant that any effects found could have been due to these factors, rather than the set and rule compatibility of each task. For example had the type A task always been "invoke "Copy formats"" it could not have been unequivocally established that type A tasks were performed more quickly than others because they represented set and rule compatible tasks, as it could be that their advantage was due to the position of "Copy formats" being more memorable than the position of other commands.

Interface configurations P, Q, R, and S are illustrated in the following figures (experimental commands are shown in italics):

FORMAT	FONT	DOCUMENT	WINDOW
MOVE TO START OF DOCUMENT	LARGER FONT SIZE	*COPY FORMATS*	*NEW WINDOW*
ZOOM WINDOW	*SMALLER FONT SIZE*	FIND FORMATS	MOVE TO END OF DOCUMENT

Experimental interface configuration P.

FORMAT	FONT	DOCUMENT	WINDOW
ZOOM WINDOW	FIND FORMATS	*MOVE TO START OF DOCUMENT*	LARGER FONT SIZE
NEW WINDOW	*SMALLER FONT SIZE*	MOVE TO END OF DOCUMENT	*COPY FORMATS*

Experimental interface configuration Q

FORMAT	FONT	DOCUMENT	WINDOW
LARGER FONT SIZE	*MOVE TO START OF DOCUMENT*	*SMALLER FONT SIZE*	ZOOM WINDOW
COPY FORMATS	MOVE TO END OF DOCUMENT	FIND FORMATS	*NEW WINDOW*

Experimental interface configuration R

FORMAT	FONT	DOCUMENT	WINDOW
COPY FORMATS	MOVE TO END OF DOCUMENT	*MOVE TO START OF DOCUMENT*	LARGER FONT SIZE
FIND FORMATS	*NEW WINDOW*	ZOOM WINDOW	*SMALLER FONT SIZE*

Experimental interface configuration S

The positioning of the experimental commands on the menus was also balanced for task type. For example, type A tasks involved invoking the first command on

the menu for configurations Q and S ("Move to the start of document" and "Copy formats" respectively), and the second command on the menu for configurations P and R ("Smaller font size" and "New window" respectively). This balance controlled for possible effects due to the position of a command on a menu. It may have been, for example, that it was easier to remember the first command than the second. There might also have been an additional time penalty associated with having to move further down a menu in order to activate the second command (although this was probably negligible in this experimental context).

Balances similar to those described in this appendix were also made in the design of the studies reported in chapters 8, 9, 10, and 11 — although there were some adaptations depending on the study (for example for differing subject numbers, or different numbers of experimental trials).

Appendix 7.2

Overview of the Experiment

The purpose of this experiment is to investigate the ease of use, or usability, of a word processing package.

During today's experimental session, which should take about an hour, you will be set a series of trials to perform, using the word processor under test. Each trial will involve activating commands in order to make alterations to a series of twelve short snippets of text.

In the second experimental session (tomorrow), which should take only a couple of minutes, you will be asked to complete one more trial.

Remember, it is the word processor that is being tested, not you, so don't worry if you have any difficulties, or cannot activate a command.

Do you have any questions ?

Appendix 7.3

Experimental Protocol

SESSION 1

0. **Describe a word processor.** If necessary (i.e. if subject has never used a word processor before) give a brief explanation of what a word processor is. Say: "A word processor is a machine which can be used to create and store text. The one you shall be using today also contains commands for manipulating and altering the text. The experiment in which you are participating is based on the invocation of these commands."

1. **Give subject the "Overview of the experiment"** to read.

2. **Administer personal details interview.** Say: "I am now going to ask you for some personal details. The details that I shall ask for are only things that are directly relevant to the experiment, and shall be treated in the strictest confidence."

3. **Give basic instruction session.** Say: "In order to perform the command invocations that you will be set during the experiment you will need to know the general principles of how to pick commands from the menus. I am now going to show you how this is done, and let you practice at it a few times, until you feel competent at it."

Show the subject the menu headings, how to open a menu, and how to activate a command. Ask the subject to open each of the commands on the "Work"[6] menu three times. When this has been done say: "Do you now feel that you are competent at the mechanics of activating a command, or do you feel that you need more practice?". If the subject wants more practice, allow him or her to continue until he or she feels competent.

---

[6]This was a menu specially created for subjects to practice with.

At the end of this session say: "That concludes the basic instruction session, we shall soon be ready to start the experimental session. Each of the commands that I shall ask you to invoke during the session is stored on one of the menus, and can be invoked by accessing the menu on which it is stored, and then going through the invocation procedure you've just practiced. You shall not need to use the "Work" menu in the experiment however, all the commands that you will be asked to invoke can be found on the other menus. Note that some menus do not have any commands on in this version of the word processor. If you pick these menus then nothing will happen. If you pick an empty menu, or one which does not contain the command that you are looking for, pressing the escape key will enable you to exit the menu and try another one."

Ask them to open "Work" menu once more, then ask them to exit it using the escape key.

4. **Start the experimental session.** Load the first file. Say: "We are now going to start the experimental session. I shall give you instructions, by telling you the name of the command that I wish you to invoke, in the form of: "Please invoke the command ... when you are ready." You will be given as many attempts as you need to invoke each of the commands set, so if you don't get it right first time please keep trying. Remember, however that it is the word processor that is under test, not you, so don't worry if you have difficulties, or get something wrong."

Note down times and errors on the experimenter's sheet.

As soon as a task is complete set the next one until the trial is over. At the end of each trial set the next one without delay.

5. **End the first session.** When all twelve trials are complete say: "That concludes today's session, thank you very much indeed for your co-operation. Please could you return tomorrow for the second session."

SESSION 2

1. **Set another experimental trial.** Load the first text file. Say: "Today I am going to ask you to complete another experimental trial."

Record time and errors

2. **Conclude the experiment.** Say: "That concludes the experiment, thank you very much indeed for your co-operation."

3. **Pay the subject.** Ensure a receipt is completed.

Appendix 8.1

Experimental configuration B

With this configuration the experimental tasks are set compatible but not rule compatible (experimental commands are shown in italics). The action-rules associated with using this interface are formalised in appendix 8.3.

FORMAT	FONT	DOCUMENT	WINDOW
Larger font size	Zoom window	*Copy formats*	Move to end document
Shapes font	Split window	Plain format	Page document
Smaller font size	Top window	Underline format	Scroll document
Type font	Open footnote window	Define format	*Move to start document*
Change fonts	*New window*	Find formats	Select whole document

Appendix 8.2

Experimental configuration C

With this configuration the experimental tasks are rule compatible but not set compatible (experimental commands are shown in italics). The action-rules associated with using this interface are formalised in appendix 8.4.

FORMAT	FONT	DOCUMENT	WINDOW
Copy formats	Zoom window	Plain format	Move to end document
Larger font size	Split window	Underline format	Page document
Shapes font	*Smaller font size*	Define format	Scroll document
Type font	Top window	*Move to start of document*	Select whole document
Change fonts	Open footnote window	Find formats	*New window*

Appendix 8.3

Formalisation of task action-rules associated with configuration B

Taskset

TASKSET = [*invoke "COPY FORMATS"*, invoke "PLAIN FORMAT", invoke "UNDERLINE FORMAT", invoke "DEFINE FORMATS", invoke "FIND FORMATS", invoke "LARGER FONT SIZE", invoke "SHAPES FONT", *invoke "SMALLER FONT SIZE"*, invoke "TYPE FONT", invoke "CHANGE FONTS", invoke "MOVE TO END OF DOCUMENT", invoke "PAGE DOCUMENT", invoke "SCROLL DOCUMENT", *invoke "MOVE TO START DOCUMENT"*, invoke "SELECT WHOLE DOCUMENT", invoke "ZOOM WINDOW", invoke "SPLIT WINDOW", invoke "TOP WINDOW", invoke "OPEN FOOTNOTE WINDOW", *invoke "NEW WINDOW"*]

Schema sets (agent = designer)

SET W = [*COPY FORMATS*, PLAIN FORMAT, UNDERLINE FORMAT, DEFINE FORMATS, FIND FORMATS]

SET X = [LARGER FONT SIZE, SHAPES FONT, *SMALLER FONT SIZE*, TYPE FONT, CHANGE FONTS]

SET Y = [MOVE TO END OF DOCUMENT, PAGE DOCUMENT, SCROLL DOCUMENT, *MOVE TO START DOCUMENT*, SELECT WHOLE DOCUMENT]

SET Z = [ZOOM WINDOW, SPLIT WINDOW, TOP WINDOW, OPEN FOOTNOTE WINDOW, *NEW WINDOW*]

Rule schemata (agent = designer)

to invoke W -> Open DOCUMENT menu + select W
to invoke X -> Open FORMAT menu + select X
to invoke Y -> Open WINDOW menu + select Y
to invoke Z -> Open FONT menu + select Z

Schema sets (agent = user)

SET P = [*COPY FORMATS*, PLAIN FORMAT, UNDERLINE FORMAT, DEFINE FORMATS, FIND FORMATS]

SET Q = [LARGER FONT SIZE, SHAPES FONT, *SMALLER FONT SIZE*, TYPE FONT, CHANGE FONTS]

SET R = [MOVE TO END OF DOCUMENT, PAGE DOCUMENT, SCROLL DOCUMENT, *MOVE TO START DOCUMENT*, SELECT WHOLE DOCUMENT]

SET S = [ZOOM WINDOW, SPLIT WINDOW, TOP WINDOW, OPEN FOOTNOTE WINDOW, *NEW WINDOW*]

Rule schemata (agent = user)

to invoke P -> Open FORMAT menu + select P
to invoke Q -> Open FONT menu + select Q
to invoke R -> Open DOCUMENT menu + select R
to invoke S -> Open WINDOW menu + select S

Set compatibility

P matches W
Q matches X
R matches Y
S matches Z

Rule compatibility

Rules for no tasks match

Appendix 8.4

Formalisation of task action-rules associated with configuration C

Taskset
TASKSET = [*invoke "COPY FORMATS"*, invoke "PLAIN FORMAT", invoke "UNDERLINE FORMAT", invoke "DEFINE FORMATS", invoke "FIND FORMATS", invoke "LARGER FONT SIZE", invoke "SHAPES FONT", *invoke "SMALLER FONT SIZE"*, invoke "TYPE FONT", invoke "CHANGE FONTS", invoke "MOVE TO END OF DOCUMENT", invoke "PAGE DOCUMENT", invoke "SCROLL DOCUMENT", *invoke "MOVE TO START DOCUMENT"*, invoke "SELECT WHOLE DOCUMENT", invoke "ZOOM WINDOW", invoke "SPLIT WINDOW", invoke "TOP WINDOW", invoke "OPEN FOOTNOTE WINDOW", *invoke "NEW WINDOW"*]

Schema sets (agent = designer)
SET W = [*COPY FORMATS*, LARGER FONT SIZE, SHAPES FONT, TYPE FONT, CHANGE FONTS]

SET X = [ZOOM WINDOW, SPLIT WINDOW, *SMALLER FONT SIZE*, TOP WINDOW, OPEN FOOTNOTE WINDOW]

SET Y = [PLAIN FORMAT, UNDERLINE FORMAT, DEFINE FORMAT, *MOVE TO START DOCUMENT*, FIND FORMATS]

SET Z = [MOVE TO END DOCUMENT, PAGE DOCUMENT, SCROLL DOCUMENT, SELECT WHOLE DOCUMENT, *NEW WINDOW*]

Rule schemata (agent = designer)
to invoke W -> Open FORMAT menu + select W
to invoke X -> Open FONT menu + select X
to invoke Y -> Open DOCUMENT menu + select Y
to invoke Z -> Open WINDOW menu + select Z

Schema sets (agent = user)
SET P = [*COPY FORMATS*, PLAIN FORMAT, UNDERLINE FORMAT, DEFINE FORMATS, FIND FORMATS]

SET Q = [LARGER FONT SIZE, SHAPES FONT, *SMALLER FONT SIZE*, TYPE FONT, CHANGE FONTS]

SET R = [MOVE TO END OF DOCUMENT, PAGE DOCUMENT, SCROLL DOCUMENT, *MOVE TO START DOCUMENT*, SELECT WHOLE DOCUMENT]

SET S = [ZOOM WINDOW, SPLIT WINDOW, TOP WINDOW, OPEN FOOTNOTE WINDOW, *NEW WINDOW*]

Rule schemata (agent = user)
to invoke P -> Open FORMAT menu + select P
to invoke Q -> Open FONT menu + select Q
to invoke R -> Open DOCUMENT menu + select R
to invoke S -> Open WINDOW menu + select S

Set compatibility
No sets match

Rule compatibility
Rules match for: *invoke "COPY FORMATS"*, *invoke "SMALLER FONT SIZE"*, *invoke "MOVE TO START DOCUMENT"*, *invoke "NEW WINDOW"*

Rules do not match for any other tasks

Appendix 8.5

Overview of the experiment

The purpose of this experiment is to investigate the ease of use, or usability, of a word processor.

During the experimental session, which should take about twenty minutes in total, you will be set a series of tasks to perform, using the word processor under test. Each task will involve invoking a word processing command. These commands will be located on menus.

Remember, it is the word processor that is being tested, not you, so don't worry if you have any difficulties, or cannot complete a task.

Do you have any questions ?

Appendix 8.6

Experimental Protocol

0. **Describe a word processor.** If necessary (i.e. if subject has never used a word processor before) give a brief explanation of what a word processor is. Say:"A word processor is a machine which can be used to create and store text. The one you shall be using during today's experiment also contains commands for manipulating and altering the text. The experiment in which you are participating is based on the use of these commands."

1. Give subject the "Overview of the experiment" to read.

Answer any general questions, but not any pertaining to the whereabouts of particular command names.

2. Administer "Personal details interview". Say: "I am now going to ask you for some personal details. The details that I shall ask for are only things that are directly relevant to the experiment, and shall be treated in the strictest confidence."

3.Give basic instruction session. Say: "In order to perform the command invocations that you will be set during the experiment you will need to known the general principles of how to pick commands from the menus. I am going to show you how this is done, and let you practice at it a few times, until you feel competent at it."

Show the subject the menu headings, and how to invoke a command. Ask the subject to invoke each of the commands on the "Work"[7] menu three times. When they have done this say: "Do you now feel that you are competent at the mechanics of activating a command, or do you feel that you need more practice?" If the

[7]This was a menu specially created for subjects to practice with.

subject requires more practice, allow him or her to continue until he or she feels competent.

At the end of this session say:"That concludes the basic instruction session, we shall soon be ready to start the experimental session. Each of the commands that I shall ask you to invoke during the session is stored on one of the menus, and can be accessed by going through the invocation procedure which you have just practiced. You shall not need to use the "Work" menu in the experiment however, all the commands that you shall be asked to invoke can be found on one of the other menus. Note that, in this version of the word processor, some menus do not contain any commands. If you pick one of these menus then nothing will happen.

If you pick an empty menu, or one which does not contain the command that you are looking for, pressing the escape key will enable you to exit the menu and try another one."

Ask the subject to open the "Work" menu once more, and then to exit it using the escape key.

4. **Start the experimental session.** Say: "We are now going to start the experimental session. I shall give you instructions by telling you the name of the command that I wish you to invoke, in the form of: "Please invoke the command..... when you are ready". You will be given as many attempts as you need to invoke each command. Remember, however, that it is the word processor under test not you, so don't worry if you have difficulties, or get some things wrong."

Note down times and errors on the experimenter's sheet.

As soon as a task is complete set the next one without delay, until the experiment is over.

5. **End the session.** When all tasks are complete say: "That concludes the experiment, thank you very much indeed for your co-operation."

Pay subjects and ensure that they complete a receipt.

Appendix 9.1

Overview of the Experiment

The purpose of this experiment is to investigate the ease of use, or usability, of an iconic interface. The interface simulates a file handling package, designed to enable a range of manipulations to be performed on different types of files.

During the experimental session you will be set a series of tasks to perform, using the interface under test. Each task will involve performing a simulated file manipulation. This will be done using the icons on the interface.

Remember, it is the interface that is being tested, not you, so don't worry if you have any difficulties, or cannot complete a task.

Do you have any questions ?

Appendix 9.2

Experimental Protocol

1. Give subject "Overview of the experiment" to read.

Answer any general questions, but not any pertaining to the meanings of particular icons.

2. Administer "Personal details interview". Say: "I am now going to ask you for some personal details. The details that I shall ask for are only things that are directly relevant to the experiment, and shall be treated in the strictest confidence."

3.Give basic instructions session. Say: "In order to perform the tasks that you will be set during the experiment you will need to known the general principles of how such tasks are performed. I am going to demonstrate this to you, and then let you practice at it a few times, until you feel competent at it."

Perform a sample task on the demonstration screen, then let the subject have a try.

At the end of this session say:"That concludes the basic instruction session, we shall soon be ready to start the experimental session.

4. Start the experimental session. Load the experimental interface. Say: "We are now going to start the experimental session. I shall give you instructions by telling you the task that I wish you to perform. If you have any difficulties or get anything wrong keep trying. Remember, however, that it is the interface under test not you, so don't worry if you have difficulties, or get some things wrong."

Note down times and errors on the experimenter's sheet.

As soon as a task is complete set the next one without delay, until the experiment is over.

If subjects are having problems with the physical mechanics of performing tasks (e.g. if they click on things in the wrong order) help them with this, but don't give any assistance if asked about icons' functions.

5. Conclude experiment. When all tasks are complete say: "That concludes the experiment, thank you very much indeed for your co-operation."

Appendix 10.1

Overview of the Experiment

This experiment is an investigation of the usability of a simulated document editing system.

As a participant in this experiment you shall, firstly, be given an introduction to the editor, and given the opportunity to practice on some "mock" tasks.

You shall then be asked to perform a number of tasks with the simulated editor, during an experimental session which should last about an hour and a half.

Remember, it is the editor that is being investigated, not you, so don't worry if you have any difficulties or get things wrong.

Tomorrow, you shall be set a few further tasks, however these should only take a few minutes.

Do you have any questions?

Appendix 10.2

Introduction to Simulated Document Editor

The interface that you shall be using is a simulation of an editor, which allows operations to be carried out which manipulate text and tables, stored in a document.

The types of operation fall into four categories:

DELETING
FORMATTING
INSERTING
SAVING

In order to perform a manipulation you must type a command into the command entry field.

Command shall consist of two words separated by a space. One of these words represents the manipulation to be performed (i.e. Deleting, Formatting, Inserting, or Saving), and the other the item upon which the manipulation is to be performed (i.e. text, or table).

After you have typed in a command the editor searches through the document until either it finds a routine for performing the specified operation, or concludes that you have typed in an incorrect command name — in which case it shall reach the end of the document without doing anything.

The editor also includes a help facility, which enables you to look up commands that you are unsure of.

I shall now give you an opportunity to practice with the editor, doing some "mock" tasks, in order that you might get a better idea of how it works.

Do you have any questions?

Appendix 10.3

Experimental Protocol

SESSION 1

1. **Give subject the "Overview of the experiment" to read.** Answer any "general" questions, but none pertaining to command names.

2. **Administer personal details interview.** Say: "I am now going to ask you for some personal details. The details that I shall ask for are only things that are directly relevant to the experiment, and shall be treated in the strictest confidence."

3. **Give subject "Introduction to the simulated document editor" to read.** Say: Before we get under way with the experiment I shall give you an introduction to the simulated editor in order that you might gain some idea of how it works. Firstly, here is a written introduction for you to read.

Give the subject the written introduction.
Answer any "general" questions, but none pertaining to the command names.

4. **Set subject some mock tasks including using the help system.** Say: "Now that you have read an introduction to the interface I shall talk you through some practice tasks."

Set the subject five mock tasks including using help — guide them through any difficulties or errors they may have. Tell the subject the exact name of the command she is to enter.

After the five examples, ask the subject if they feel confident with the "mechanics" of command entry and using the help system. If she is then go on to experimental session - if not then set some more mock commands.

5. **Start experimental session.** Leave command entry box open from practice session. Say: "We are now going to start the experimental session. I shall give you instructions as we go along, by telling you the manipulation which I wish you to perform. I shall give you as long as you need to complete each of the tasks that I shall set you. Feel free to consult the help facility at any time during the experiment, but don't worry about entering an incorrect command - if your not sure of a command it may be worth a guess, as continual use the help facility is time consuming.

As you are working through the tasks I shall be sitting here writing things down, and timing you. Try not to be put off by either. I am writing about the editor, not you, and I am only timing in order that I might get average figures for the length of time users take over particular tasks. Remember, that it is the editor, not you, under test, so don't worry if you have difficulties, or get something wrong."

Activate editor, and open command box for subjects.
Note down time, errors, and help consultations on the experimenter's sheet.

As soon as a task is complete, set the next one until the experimental session is over.

Give help with any problems concerning the "mechanics" of the editor — but not with command names.

6. **Conclude first session.** Say: "That concludes todays experimental session. Please remember to return tomorrow at the allotted time."

SESSION 2

1. **Set an experimental trail.** Say: "Today, I an going to set you a few more tasks, like those you did yesterday."

Set the trial.

2. **Conclude the experiment.** Say: "That concludes the experiment. Thank you very much indeed for your participation."

3. **Pay the subject.** Ensure a receipt is completed.

Appendix 11.1

Example of Changes Made to an Experimental Letter

In this appendix a letter is depicted before (this page) and after (next page) the changes made in the first experimental trial. The changes are as follows: the sender's address has been centred, the sender's address has been put into "Bold" style, and, after moving the cursor to the end of the letter, the date has been added

Smith's Nurseries
The Old Farmhouse
Blackthorpe
Lothian

Mr R. McLeod
12 Northpark Way
Baltonsleigh
Strathclyde

Dear Mr McLeod,

Thank you very much for your application to work as a gardener at out nursery. Unfortunately, however, all posts are now filled.

Yours sincerely,

Andrew Robertson (Manager)

Smith's Nurseries
The Old Farmhouse
Blackthorpe
Lothian

Mr R. McLeod
12 Northpark Way
Baltonsleigh
Strathclyde

Dear Mr McLeod,

Thank you very much for your application to work as a gardener at out nursery. Unfortunately, however, all posts are now filled.

Yours sincerely,

Andrew Robertson (Manager)

10/11/1992

Appendix 11.2

Overview of the Experiment

Thank you for agreeing to participate in this experiment, the purpose of which is to investigate the usability of a word processing package running on a Macintosh personal computer.

Before the experimental session starts, you will be given an introduction to the Macintosh operating environment, via a brief training session on a statistics package.

You will then be asked to edit a series of letters using the word processor under test. For each letter a number alterations are required — you will be informed as to what these are at the time. The whole experiment should last for about an hour.

Remember, it is the word processor that is being tested not you, so don't worry if you have any difficulties or make errors.

Do you have any questions?

Appendix 11.3

Practice Tasks

Subjects were talked through the following practice tasks with the "Statview" statistics package

1. Assigning variables — done via "Quick assignment" command on "VARS" menu.

2. Calculating means — done via "Means, standard dev. etc." command on "DESCRIBE" menu.

3. Viewing via a barchart — done via "Barchart" command on "VIEW" menu.

4. Viewing via a table — done via a combination of the "apple" key and the letter "T" on the keyboard.

5. Assigning variables — done via "Quick assignment" command on "VARS" menu.

6. Performing an Anova — done via "Anova" command on "COMPARE" menu.

7. Sorting the data — done via "Sort" command on "TOOLS" menu.

8. Changing the font — done via "Helvetica" command on "FONT" menu.

9. Saving the changes — done via "Save" command on "FILE" menu.

10. Quitting from the program — done via a combination of the "apple" key and the letter "Q" on the keyboard.

Appendix 11.4

Experimental Protocol

1. Describe a word processor. If necessary (i.e. if subject has never used a word processor before) give a brief explanation of what a word processor is.

Say: "A word processor is a machine which can be used to create and store text. The one which you shall be using today also contains commands for manipulating and altering the text. You will be required to use these commands, in order to perform the tasks set in the experiment."

2. Give the subject "Overview of the experiment" to read.

3. Administer "Personal details interview".

Say: "I am now going to ask you for some personal details. The details I shall ask for are only those which may be relevant in this experimental context, and shall be treated in the strictest confidence."

4. Give introductory session using Statview.

Say: "In order to give you an idea about how the Macintosh operating system works, I shall now introduce you to a statistics package. Although the functions on this package will be largely different from those on the word processor, using it should give you a general idea about the principals behind Macintosh interfaces."

See introductory session task sheet.

5. **Start the experimental session.** Load the first file. Say: "We are now going to start the experimental session. I shall give you instructions, by telling you the operations that I wish you to perform, in the form of: "Please could you..... when you are ready. If you are having trouble with an operation, you may consult the help system, which you will find on the window menu. However, if you are in doubt as to how to perform an operation, I would encourage you to have a guess before resorting to using the help system — remember we are testing the word processor's usability, so any errors that you make are a reflection on the word processor, not on you."

Explain how to access help files

Say: "Throughout the session I shall be writing down any usability problems that arise, and the time needed to complete each task — these will be used as indicators of the word processors performance."

Inform subject of the operations to be performed as the session progresses

Note down times, errors, help consultations, and experimenter interventions on the experimenters sheet.

As soon as each task is complete set the next all tasks have been completed.

6. **Conclude the session** - after the completion of all experimental and filler tasks.

Say: "That concludes the experiment thank you very much for your participation."

7. **Pay the subject.** Ensure that they complete a receipt.

Appendix 12.1

Overview of the Experiment

The purpose of this experiment is to compare how demanding it is to learn the "WORD" word processing package on the Macintosh computer. The experiment will take about an hour and a half in total, and involves performing tasks with the word processor.

The tasks that you will be given all involve converting letters, which will appear on the computer into a standard form — the experimenter will explain the tasks in more detail when we are ready to start. If you get stuck during the experiment (being stuck = you do not know what to do for more than about two minutes, or feel like asking someone else what to do) the experimenter shall be able to give you a limited amount of help on request — however the way in which the experimenter intervenes will depend on how far you got before getting stuck. Remember, it is the word processor that is being tested not you, so don't worry if you have any difficulties or get things wrong.

Before the experiment begins you will be given a personal details interview, in which you will be asked for details relevant to the experiment. Your responses to this, and data gained from your experimental session will be treated in the strictest confidence.

Appendix 12.2

Examples of a Letter in "Plain" and "Standardised" Forms

In this appendix two versions of the same letter are shown. On this page, the letter is in its original "plain" form, and on the next page it is shown in its "standardised" form after completion of the experimental sub-tasks. These sub-tasks are listed in appendix 12.3.

Aberdeen School of Executive Learning
234 Cumberland Square
Aberdeen

23rd November 1990

Mr A. Walsh
Glasgow Office Training Ltd.
10 Portwood Terrace
Glasgow

Dear Mr Walsh,

I would like to be put on your mailing list for information about future courses in word processing.

Unfortunately I am unable to attend the one in January due to business commitments.

Yours sincerely,

Robert Taylor.

Re: Courses in word processing

Aberdeen School of Executive Learning
234 Cumberland Square
Aberdeen

23rd November 1990

Mr A. Walsh
Glasgow Office Training Ltd.
10 Portwood Terrace
Glasgow

Dear Mr Walsh,

Re: Courses in word processing

I would like to be put on your mailing list for information about courses in word processing.

Unfortunately I am unable to attend the one in January due to pressing business commitments.

Yours sincerely,

Robert Taylor.

Appendix 12.3

How to Standardise a Letter

1. Open the file containing the letter to be standardised. Files are named after the person to whom the letter is being sent (Eg. if a letter was being sent to Ms Francis it would be in a file called "Ms Francis").

2. Move the sender's address to the top centre of the page.

3. Put the senders address into shadow format.

4. Increase the size of the lettering of the senders address to 18 point font.

5. Move the date to the extreme right of the page.

6. Put the receivers address into bold format.

7. Scroll down so that the line of the letter containing the receivers name (i.e.. the line that says: Dear Mr/Ms .....) is right at the top of the screen.

8. Delete the unnecessary word in the letter. This word is one that appears in the original letter, but not in the standard letter. In the copy of the standard letters that you have its position is marked by a cross.*

9. Insert the missing word into the letter. This word is one that appears in the standard letter, but is not in the original letter. It is ringed in your copy of the standard letter.*

10. Move the last line of the original letter into the place indicated on your copy of the standard letter.

11. Save the changes that you have made to the letter.

12. Close the file containing the letter.

* You will need to use the keyboard for part of these tasks.

Appendix 12.4

Experimental Protocol

1. Give subjects "Overview of the experiment" to read.

2. Administer personal details interview. Read this out to the subjects and fill it in myself.

3. Give the basic instruction session.

4. Issue subjects with the manuals.
Say: "These are the manuals that are provided with the word processor. Feel free to consult them whenever you wish, during the experiment."

5. Advise subjects about getting help.
Say: "If you get stuck you may ask me for help. Getting stuck means: either not knowing what to do for longer than about two minutes, or feeling as if you need to ask someone what to do. The level at which I can intervene will depend on how far you got with the task before getting stuck."

6. Introduction to session.
Say: "In a moment I am going to give you the list of tasks, and allow you to get started with the word processor. You may feel that you wish to start on the tasks straight away, or you may prefer to spend some time familiarising yourself with the word processor first — it is entirely up to you. If you do wish to practice with the word processor first then please use the empty practice file that has been provided for this."

7. Give the subjects versions of the experimental letter before and after conversion and the instructions for converting the original letters into standard form.

Say: "You have been given two copies of a letter. One copy shows the letter in it's original form, as it will appear on the computer, and the other shows the same letter in standard form — the red pen marks indicating what changes have been made. Your task is to convert the letter as it appears on the computer into standard form. With the copies of the letter there is a sheet of instructions that tell you how to convert the letter from original to standard form."

8. Allow the subjects time to read the instructions.

Say: "You may now spend a couple of minutes reading the instructions so that you understand what you will have to do. Whilst you are doing that I shall get the computer ready for the experiment."

Time how long subjects spend with the instructions.

If they have not indicated that they are ready to proceed after 3 minutes then ask if they are happy with the instructions. If they are not then allow them another 2 minutes before asking again up to a maximum total of 9 minutes. If after 9 minutes they are still not happy then allow them 1 more minute before moving on.

Do not answer any questions relating to the use of the word processor, but do answer any other questions about the instructions, so that they are clear what changes they have to affect to create the standard letter.

9. **Tell the subjects that they may now start the session.**
Say: "In a moment you may start the session. Remember you may start on the tasks when you wish. If you need help please remember to fill in the incident diary first. Remember this is a test of the word processor, not of you, so don't worry about any problems or difficulties you may have. You may start when you are ready."

Fill in free session sheet if necessary
Fill in the task score sheet (timing every sub-task and noting any problems or errors).
If help is required then follow the help procedure and record the level of help that was given on the task score sheet.

After all the task sheet has been completed stop the subjects.

10. **Thank subjects for their participation.**

REFERENCES

ALLEN, R.B., and SCERBO, M.W., 1983. 'Details of command language keystrokes'. *ACM Transactions on Office Information Systems*, Vol. 1, No. 2. pp 159-178.

BANNON, L.J. and BODKER, S., 1991. 'Beyond the interface: encountering artifacts in use.' In J.M. Carroll (ed), *Designing Interaction: Psychology at the Human-Computer Interface.* (Cambridge: Cambridge University Press). pp 227-253.

BARNARD, P.J., HAMMOND, N.V., MORTON, J., LONG, J. and CLARK, I.A., 1981. 'Consistency and compatibility in human-computer dialogue'. *International Journal of Man-Machine Studies*, Vol. 5. pp 87-134.

BAXTER, I. and OATLEY, K., 1991. 'Comparing the learnability of spreadsheets'. *Behaviour and Information Technology*, Vol. 10, No. 6.

BENNETT, J.L., 1984. 'Managing to meet usability requirements: establishing and meeting software development goals'. In J. Bennett and D. Care (eds.), *Visual Display Terminals.* pp 161-184.

BLACK, J.B. and MORAN, T.P., 1982. 'Learning and remembering command names'. In *CHI '82 proceedings.* (New York: ACM). pp 8-11.

BOOTH, P.A., 1989. *An Introduction to Human-Computer Interaction.* (Hove and London: LEA Publishers).

BROOKE, J., BEVAN, N., BRIGHAM, F., HARKER, S., and YOUMANS, D., 1990. 'Usability statements and standardisation — work in progress in ISO'. In D. Diaper, D. Gilmore, G. Cockton and B. Shackel (eds.), *Human-Computer Interaction — INTERACT '90.* (North Holland: Elsevier).

CARD, S.K., ENGLISH, W.K., and BURR, B., 1978. 'Evaluation of mouse rate controlled isometric joystick, step keys, and text keys for text selection on a CRT'. *Ergonomics*, Vol. 21. pp 601-613.

CARD, S.K., MORAN, T.P., and NEWELL, A., 1983. *The Psychology of Human Computer Interaction.* (Hillsdale, New Jersey: LEA Publishers).

CARROLL, J.M., 1980. "Naming' as a mapping between n-dimensional geometries'. *IBM Research Report*, RC 8596.

CARROLL, J.M., 1982. 'Learning, using and designing filenames and command paradigms.' *Behaviour and Information Technology*, Vol. 1, No. 4. pp 327-346.

CROSSMAN, E.R.F.W., 1959. 'A theory of the acquisition of speed skill'. *Ergonomics*, Vol. 2. pp 153-166.

DANIEL, W.W., 1977. *Introductory statistics with applications.* (Boston: Houghton Mifflin Company).

DRAPER, S.W., 1985. 'The nature of expertise in UNIX'. In B. Shackel (ed.), *Human-Computer Interaction — INTERACT '84.* (North Holland: Elsevier)

DRAPER, S.W., 1993. 'The notion of task in HCI'. In *INTERCHI '93 Proceedings.* (New York: ACM).

EASON, K.D., 1984. 'Towards the experimental study of usability'. *Behaviour and Information Technology*, Vol.3, No. 2. pp 133-145.

ELMES, D.G., KANTOWITZ, B.H., and ROEDIGER, H.L., 1981. *Research Methods in Psychology.* (St Paul: West Publishing Company).

EDGERTON, E.A., and DRAPER, S.W., 1993. 'A comparison of the feature checklist and the open response questionnaire in HCI evaluation'. *Computing Science Research Report* GIST-1993-1. (University of Glasgow: Department of Computing Science).

FISHER, J., 1991. 'Defining the novice user'. *Behaviour and Information Technology*, Vol. 10, No. 5, pp 437-441.

FITTS, P.M. and SEEGER, C.M., 1953. 'S-R compatibility: spatial characteristics of stimulus and response codes'. *Journal of Experimental Psychology*, Vol. 46. pp 199-210.

FURNAS, G.W., LANDAUER, T.K., GOMEZ, L.M., and DUMAIS, S.T., 1983. 'Statistical semantics: analysis of the potential performance of key-word information systems'. *The Bell System Technical Journal*, Vol. 62, No. 6. pp 1753-1805.

GILB, T., 1981. 'Design by objectives'. Unpublished manuscript.

GRUDIN, J., 1989. 'The case against user interface consistency'. *Communications of the ACM*, Vol. 32, No. 10. pp 1164-1173.

HOWES, A. and PAYNE, S.J., 1990. 'Display-based competence: towards user models for menu-driven interfaces'. *International Journal of Man-Machine Studies*, Vol. 33. pp 637-655.

HOPPE, H.U., TAUBER, M., and ZIEGLER, J.E., 1986. 'A survey of models and formal description methods in HCI with example applications'. *Esprit Project 385 HUFIT Report* B3.2a. (Fraunhofer-Institute).

JORDAN, P.W., 1990. 'Driving and stereo use: a human factors evaluation'. *MSc (Eng) report.* (University of Birmingham: Centre for Ergonomics and Operational Research).

JORDAN, P.W., 1992. 'Ergonomic design for vehicle users of new technology interfaces'. *Engineering Designer*, Vol. 8, No. 3. pp 16-20.

JORDAN, P.W., 1992a. 'Claims for direct manipulation interfaces investigated'. *Industrial Management and Data Systems*, Vol. 92, No. 8. pp 3-6.

JORDAN, P.W., 1992b. 'Making friends with the user'. *Professional Engineering*, Vol. 5, No. 9, pp 22-23.

JORDAN, P.W., 1993. 'Methods for user interface performance measurement'.In E.J. Lovesey (ed), *Contemporary Ergonomics 1993.* (London: Taylor and Francis).

JORDAN, P.W., DRAPER S.W., MacFARLANE K.K., and McNULTY, S-A., 1991. 'Guessability, learnability and experienced user performance'. In D. Diaper and N. Hammond (eds.), *People and Computers VI.* (Cambridge: Cambridge University Press). pp 237-245.

JORDAN, P.W. and JOHNSON, G.I., 1991. 'The usability of remote control for in-car stereo operation'. In E.J. Lovesey (ed), *Contemporary Ergonomics 1991.* (London: Taylor and Francis). pp 400-406.

JORDAN, P.W. and KERR, K.C., 1993. 'Usability testing in the 'real world': evaluating a multi-function telephone system'. In In E.J. Lovesey (ed.), *Contemporary Ergonomics 1993.* (London: Taylor and Francis).

JORDAN, P.W. and OATLEY, K., 1992. 'User interface performance measurement. Methods for assessing human-machine interaction'. *Video script*. (University of Glasgow: Department of Psychology).

JORDAN P.W., and O' DONNELL, P.J., 1992. 'Quantifying guessability, learnability and experienced user performance'. In E.J. Lovesey (ed.), *Contemporary Ergonomics 1992.* (London: Taylor and Francis). pp 404-410.

JORDAN, P.W., and O'DONNELL, P.J., 1992a. 'The index of interactive difficulty'. In E.J. Lovesey (ed.), *Contemporary Ergonomics 1992.* (London: Taylor and Francis). pp 397-402.

JOSEPH, C., 1991. 'There's something 'Gooey' on my screen!'. In *The Independent* (newspaper), 11 February 1991. pp 16-17.

KARAT, J., FOWLER, P., and GRAVELLE, M., 1987. 'Evaluating user interface complexity'. In H-J. Bullinger and B. Shackel (eds.), *Human-Computer Interaction — INTERACT '87.* (North Holland: Elsevier). pp 489-495.

KELLOGG, W.A., 1987. 'Conceptual consistency in the user interface: effects on user performance'. In H-J. Bullinger and B. Shackel (eds.), *Human-Computer Interaction — INTERACT '87.* (North Holland: Elsevier). pp 389-394.

KIERAS, D.E. and POLSON, P.G., 1985. 'An approach to the formal analysis of user complexity'. *International Journal of Man-Machine Studies*, Vol. 22. pp 365-394.

KIRAKOWSKI, J., and CORBETT, M., 1988. 'Measuring user satisfaction'. In D.M. Jones and R. Winder (eds.), *People and Computers IV.* (Cambridge: Cambridge University Press). pp 329-338.

KOLERS, P.A., 1975. 'Memorial consequences of automatised encoding'. *Journal of Experimental Psychology: Human Learning and Memory*, Vol. 1, No. 6. pp 687-701.

LANDAUER, T.K., 1987. 'Relations between cognitive psychology and computer system design'. In J.M. Carroll (ed.), *Interfacing Thought.* (London: MIT Press). pp 1-25.

LANDAUER, T.K., CALOTTI, K.M., and HARTWELL, S., 1983. 'Natural command names and initial learning: a study of text editing terms'. *Communications of the ACM*, Vol. 26, No. 7. pp 495-503.

LEE, A.Y., POLSON, P.G., and BAILEY, W.A., 1989. 'Learning and transfer of measurement tasks'. In *CHI '89 Proceedings.* (New York: ACM). pp 115-120.

LEWIS, C., HAIR, D.C., and SCHOENBURG, V., 1989. 'Generalisation, consistency, and control'. In *CHI '89 Proceedings.* (New York: ACM) pp 1-5.

MAISSEL, J., 1990. 'Development of a methodology for icon evaluation'. NPL Report DITC 159/90. (Teddington: National Physical Laboratory).

MARSHALL, C.J., NELSON, C., and GARDINER, M.M., 1987. 'Design guidelines'. In M. M. Gardiner and B. Christie (eds.), *Applying Cognitive Psychology to User-Interface Design.* (Chichester: Wiley).

MAYES, J.T., DRAPER, S.W., McGREGOR, A.M., and OATLEY, K., 1988. 'Information flow in the user interface'. In D.M. Jones and R. Winder (eds.), *People and Computers IV.* (Cambridge: Cambridge University Press).

MAZUR, J. and HASTIE, R., 1978. 'Learning as accumulation: A re-examination of the learning curve'. *Psychological Bulletin*, Vol. 85. pp 1256-1274.

MOHAGEG, M.F., 1991. 'Object-oriented versus bit-mapped graphics interfaces: performance and preference differences for typical applications.' *Behaviour and Information Technology*, Vol.10, No. 2. pp 121-147.

MORAN, T.P., 1981. 'The command language grammar: A representation for the user interface of interactive computer systems'. *International Journal of Man-Machine Studies*, Vol 15. pp 3-50.

MORIN, R.E and GRANT, D.A., 1955. 'Learning and performance on a key-pressing task as function of the degree of spatial stimulus-response correspondence'. *Journal of Experimental Psychology*, Vol. 49, No. 1. pp 39-47.

NEWELL, A. and ROSENBLOOM, P.S., 1981. 'Mechanisms of skill and the power law of practice'. In J.R. Anderson (ed.), *Cognitive Skills and Their Acquisition.* (Hillsdale, New Jersey: LEA Publishers). pp 1-55.

NORMAN, D.A., 1988. *The Psychology of Everyday Things.* (New York: Basic Books).

NORMAN, D.A., DRAPER, S.W., and BANNON, L.J., 1986. 'Glossary'. In D.A. Norman and S.W. Draper (eds.), *User Centred System Design.* (Hillsdale, New Jersey: LEA Publishers). pp 487-497.

NORMAN, D.A. and FISHER, D., 1982. 'Why alphabetic keyboards are not easy to use: keyboard layout doesn't much matter' *Human Factors*, Vol. 24. pp 509-519.

O'DONNELL, P.J., SCOBIE, G. and BAXTER, I., 1991. 'The use of focus groups as an evaluation technique in HCI'. In D. Diaper and N. Hammond (eds.), *People and Computers VI.* (Cambridge: Cambridge University Press). pp 211-224.

PAYNE, S.J., 1985. 'Task-action grammars'. In B. Shackel (ed.), *Human-Computer Interaction — INTERACT '84.* (North Holland: Elsevier) pp 527-532.

PAYNE, S.J., 1985a. 'Task-action grammars: The mental representation of task languages in human-computer interaction'. *Doctoral dissertation.* (University of Sheffield).

PAYNE, S.J. and GREEN, T.R.G., 1983. 'The user's perception of the interaction language: a two level model'. In *CHI '83 Proceedings.* (New York: Academic Press). pp 202-206.

PAYNE, S.J. and GREEN, T.R.G., 1984. 'Organisation and learnability in computer languages'. *International Journal of Man-Machine Studies,* Vol. 21. pp 7-18.

PAYNE, S.J., and GREEN, T.R.G., 1986. 'Task-action grammars: a model of the mental representation of task languages'. *Human-Computer Interaction,* Vol. 2. pp 93-133.

PAYNE, S.J., and GREEN, T.R.G., 1989. 'The structure of command languages: an experiment on task-action grammar'. *International Journal of Man-Machine Studies,* Vol. 30. pp 213-234.

PAYNE, S.J., and GREEN, T.R.G., 1990. 'Task-action grammar: recent developments'. In D. Diaper (ed.), *Approaches to Task Analysis.* (Cambridge: Cambridge University Press).

POLSON, P.G., 1988. 'The consequences of consistent and inconsistent interfaces'. In R. Guindon (ed.), Cognitive Science and its Application for Human-Computer Interaction. (Hillsdale, N.J.: LEA Publishers).

RAVDEN, S.J. and JOHNSON, G.I., 1989. *Evaluating Usability of Human-Computer Interfaces: a Practical Method.* (Chichester: Ellis Horwood).

REISNER, P., 1981. 'Formal grammar and human factors design of an interactive graphics system'. *IEEE Transactions on Software Engineering*, Vol. SE-7, No. 2, pp 229-240.

REISNER, P. 1990. 'What is inconsistency?'. In D. Diaper et al. (eds.), Human-Computer Interaction — INTERACT '90. (North Holland: Elsevier). pp 175-181.

ROSENBURG, J.K., 1982. 'Evaluating the suggestiveness of command names'. Behaviour and Information Technology, Vol. 1. pp 118-128.

SCAPIN, D.L., 1982. 'Generation effect, structuring and computer commands'. Behaviour and Information technology, Vol. 1. pp 401-410.

SHNEIDERMAN, B., 1987. *Designing the User Interface.* (Reading, Mass: Addison-Wesley).

SHACKEL, B., 1986. 'Ergonomics in design for usability'. In M.D. Harrison and A. Monk (eds.), *People and Computers: Designing for Usability.* (Cambridge: Cambridge University Press). pp 44-64.

SHACKEL, B., 1991. 'Usability — context, framework, definition, design, and evaluation'. In B. Shackel and S. Richardson (eds.), *Human Factors for Informatics Usability.* (Cambridge: Cambridge University Press). pp 21-37.

SMITH, S.L., and MOSIER, J.N., 1986. 'Guidelines for designing user interface software'. Report 7 MTR-10090, Esd-Tr-86-278. (Bedford, Mass.: MITRE Corporation).

STREITZ, N.A., SPIJKERS, W.A.C., and VAN DUREN. L., 1987. 'From novice to expert user: a transfer of learning experiment on different interaction modes'. In H-J. Bullinger and B. Shackel (eds.), *Human-Computer Interaction — INTERACT '87.* (North Holland: Elsevier).

SUCHMAN, L.A., 1987. *Plans and Situated Actions: the Problem of Human-Machine Communication.* (Cambridge: Cambridge University Press).

THIMBLEBY, H., 1991. 'Can humans think?'. *Ergonomics*, Vol. 34, No. 10, pp 1269-1288.

VOSSEN, P.H., SITTER, S. and ZIEGLER, J.E., 1987. 'An empirical validation of Cognitive Complexity Theory'. In In H-J. Bullinger and B. Shackel (eds.), *Human-Computer Interaction — INTERACT '87.* (North Holland: Elsevier). pp 71-75.

WHITESIDE, J., BENNETT, J., and HOLTZBLATT. K., 1988. 'Usability engineering: our experience and evolution'. In M. Helander (ed), *Handbook of Human-Computer Interaction.* (North Holland: Elsevier).