An Investigation of Object-oriented Interfaces for Human Computer Interaction

by

Carolyn Margaret Jean Selby

University College London

Submitted for the degree of Doctor of Philosophy in the University of London

August 1990
Abstract

The thesis reports an investigation of object-oriented interfaces which was undertaken to advance understanding of their implications for the user. The object-oriented interface is a comparatively recent development in user computer interface technology but, despite its popularity and reputation for ease of use, little research about it has been reported from a cognitive ergonomics perspective.

The investigation involves two activities, one analytic and one empirical, aimed at providing a characterization and assessment of object-oriented interfaces. The analytic activity specifies a framework within which object-oriented interfaces can be described and the empirical activity yields experimental findings concerning the usability of object-oriented interfaces.

First, the framework is proposed. It comprises some distinctions, definitions and a levels structure for modelling devices and users. Throughout the investigation, the structure is applied to derive particular models of devices and user behaviour. This process enables elements of the framework to be validated or modified.

On the empirical side, an observational study is conducted which compares user performance on an object-oriented interface and a non-object-oriented interface. It determines that object-oriented interfaces are easier to use and that the dimensions for classifying object-oriented interfaces have consequences for user performance.

Following the observational study, a specific device feature, syntax of the command sequence, is selected for further experimentation and a rudimentary user model is proposed. Two experiments, using simulated interfaces, are conducted to discover the effects of I/O Level and Task Level factors on syntax usability. The results of these studies advance the user model.

These studies suggest there is no inherent difference between the two command sequences tested. An alternative view is proposed which suggests that performance differences found in the observational study are due to characteristics particular to objects which were not tapped in the experimental studies. A third experiment is performed which lends some support this proposition.

The thesis concludes that:

i) an object-oriented interface can be distinguished from other interface classes in terms of features which have consequences for the user;

ii) under certain conditions, object-oriented interfaces are easier to use than their contrast and the object-oriented variant of the feature, syntax, has a facilitatory influence on user performance.
## Contents

<table>
<thead>
<tr>
<th>Title</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>2</td>
</tr>
<tr>
<td>Contents</td>
<td>3</td>
</tr>
<tr>
<td>List of Figures</td>
<td>11</td>
</tr>
<tr>
<td>List of Tables</td>
<td>13</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>14</td>
</tr>
</tbody>
</table>

### Chapter 1: Introduction

1. Introduction  
2. The Thesis as a Contribution to Cognitive Ergonomics  
3. The Topic Under Investigation: Object-oriented Interfaces  
   3.1 What is an Object-oriented Interface?  
   3.2 Why study Object-oriented Interfaces I: the emergence and market success of object-oriented interfaces  
   3.3 Why study Object-oriented Interfaces II: the apparent ease-of-use of object-oriented interfaces  
4. The Investigative Approach  
5. Organisation of the Thesis  

### Chapter 2: The Framework

1. Introduction  
2. The Need for a Framework  
3. What is a Framework?  
4. The Requirements for a Framework  
5. Proposal for the Framework  
   5.1 Scope 1: Set of Distinctions  
      5.1.1 The Real World and the Domain of Application  
      5.1.2 Device World  
      5.1.3 User World  
      5.1.4 Relationship between the Three Worlds  
   5.2 Scope 2: Definitions  
      5.2.1 Features  

3
Chapter 3: Deriving the Dimensions for the Device Structure

1. Introduction 50
2. Dimensions and the Mechanism for Classification 50
   2.1 What is a Dimension? 50
   2.2 How Dimensions Work in Classification 51
3. Observation of Extant Systems 53
   3.1 Representation of task elements 53
   3.2 Limitation of Command Set 58
   3.3 Syntax of Command Sequence 60
   3.4 Ratio between Objects and Actions 61
   3.5 The Representation of the Properties of Objects 62
   3.6 Concealing the Mechanics from the User 62
   3.7 Summary 63
4. Reviewing the Literature 63
   4.1 Object-oriented Programming 64
   4.2 Object-oriented Systems 67
   4.3 Some Direct References to Object-oriented Interfaces 68
   4.4 Miscellaneous References to the Term, 'Object-oriented' 70
   4.5 Summary 74
5. The Set of Derived Dimensions 74
   5.1 The Nature of the Interface and Ways to Specify its Object-orientedness 75
   5.2 Integrated Set of Possible Dimensions 76
6. Summary 79

Chapter 4: Validating the Dimensions: an Observational Study

1. Introduction 81
2. Rationale for Observational Study and its Objectives 81
Chapter 5: Method for Testing

1. Introduction 128
2. Rationale for Single Dimension Experiments 128
3. Selection of the Dimension, Syntax 129
   3.1 Syntax: the Findings of the Observational Study and a Rudimentary User Model 130
4. Method of Experimentation 135
   4.1 Introduction to HyperCard 136
   4.2 The Use of HyperCard to Simulate Interfaces 140
   4.3 General Description of the Interfaces Generated by HyperCard 141
5. Concluding Remarks 142

Chapter 6: Investigating the Usability of an Object First Syntax under Different I/O Level Conditions: the First Experimental Study (E1)

1. Introduction 144
2. Aims of the Study 144
3. Background 146
4. Details of the Simulated Interface 148
   4.1 Requirements for the Simulated Interface 148
   4.2 Description of the Simulation 150
5. Method 153
   5.1 Design 153
   5.2 Subjects 154
   5.3 Tasks 154
   5.4 Training 154
   5.5 Experimental Session 155
6. Results 155
   6.1 Errors 155
      6.1.1 Overall Number of Errors 156
      6.1.2 Classification of Error Types 158
      6.1.3 Errors and Trial Type 159
   6.2 Time Data 160
7. Discussion 161
   7.1 Implications of the Results 162
      7.1.1 On the Difference in Performance found between the OC and AC Groups 162
Chapter 8: The Ease-of-Use of Object-oriented Interfaces: Two Views

1. Introduction 216

2. Discrepancies between Studies 216
   2.1 The Entity View 217
   2.2 The Object Advantage View 217

3. Possible Factors Responsible for an Object Advantage 218
   3.1 Hiding 218
      3.1.1 The Relationship between the Notion of Hiding and other Concepts in the Literature
         3.1.1.1 Related Concepts in the Literature 220
         3.1.1.2 The Relationship between the above Concepts and Hiding 222
      3.1.2 The Potential Advantages of Hiding 223
      3.1.3 Hiding and Objects 224
   3.2 Differences in Representations for Knowledge about Objects and Actions: Evidence from the Psychological Literature
      3.2.1 Analytic Research in Semantic Memory 227
      3.2.2 Empirical Work in Semantic Memory 228
      3.2.3 Findings from Word Association Experiments 229
      3.2.4 Summary 230
   3.3 Interrelationships between Nouns and Verbs 230
      3.3.1 Evidence from the Associative Thesaurus 230
         3.3.1.1 Consulting the Thesaurus 231
         3.3.1.2 Problems Encountered 232
         3.3.1.3 General Findings 233
         3.3.1.4 Conclusion 235
      3.3.2 An Exploratory Study Addressing Object Action Asymmetries 235
         3.3.2.1 Details of the Experimental Testbed 235
         3.3.2.2 Design 237
         3.3.2.3 Results 237
         3.3.2.4 Conclusion 235
   3.4 General Summary Concerning Object Specific Properties 238

4. Implications if the Two Views for the User Model 239

5. Future Directions 239
Chapter 9: The Effects of Non-specific Instructions on the Usability of Alternative Syntax Structures: the Third Experimental Study (E3)

1. Introduction 241
2. Aims of the Study 242
3. Background 242
4. Details of the Simulated Interface 243
5. Method 245
   5.1 Design 245
   5.2 Subjects 246
   5.3 Environment 246
   5.4 Tasks 247
   5.5 Training 248
   5.6 Experimental Session 248
6. Results 248
   6.1 Number Correct at First Attempt 249
   6.2 Time 250
   6.3 Number of Commands 251
   6.4 Number of Pop-ups 252
   6.5 Subjects' Comments 253
7. Discussion 254
   7.1 Implications of the Results: Which View is Supported? 254
   7.2 Updating the User Model 257
8. Summary and Conclusions 262

Chapter 10: Overview

1. Introduction 265
2. Purpose of Chapter 265
3. The Approach 266
4. Characterization of Object-oriented Interfaces 266
   4.1 Overview of Output: Characterization 267
   4.2 Consideration of the Question, What is an Object-oriented Interface? 269
   4.3 Can the Device Structure Distinguish Object-oriented Interfaces from Related Types of Interface? 272
   4.4 The Relationship between Object-oriented Interfaces and Object-oriented Programming 277
   4.5 Limitations of the Work Concerning Characterization 279
List of Figures

1.1 First, current and main Macintosh applications among corporate users (MacNews/Times 1000 Survey, MacNews, July 1989) 23
1.2 Reasons why Macintoshes were purchased (MacNews/Times 1000 Survey, MacNews, July 1989) 24
2.1 The relationships between worlds 36
2.2 Classification of objects with respect to appearance and behaviour 40
2.3 General user structure 47
3.1 Pictorial representation of classification of interfaces using dimensions 52
3.2 Examples of object icons 54
3.3 Two images of the wastebin (from Apple Macintosh) 55
3.4 Example of an action icon using an arrow 56
3.5 Example of an action icon which shows two states of an object 56
3.6 An example of a limited option set, presented as a menu 58
3.7 Examples of two dialogue structures 60
4.1 Illustration of the Lisa screen layout 86
4.2 Illustration of the TopView screen layout 89
4.3 Example of an unintended action => object syntax sequence 121
5.1 Rudimentary User Structure 132
5.2 Rudimentary User Model (TopView subject) 134
5.3 Examples of HyperCard objects 138
6.1 Initial screen display for Pick & Point interface 151
6.2 Initial screen display for Command Line interface 152
6.3 Error message for Command Line interface 152
6.4 Subject groups 153
6.5 Graph of overall number of errors by group 157
6.6 User Structure for E1 (Pick & Point conditions) 167
6.7 Example of User Model for Pick & Point condition (a => o) 169
6.8 User Structure for E1 (Command Line conditions) 171
6.9 Example of User Model for Command line condition, action first syntax 173
6.10 Example of User Model for Command line condition, object first syntax 174

7.1 Example of an 'option' screen 186
7.2 Graph of mean error and sub-optimal response scores (S17-32) 199
7.3 Graph of mean times to select first command (S17-32) 201
7.4 User structure for E2 207
7.5 Behavioural example of User Model for a consistent trial 209
7.6 Behavioural example of User Model for an inconsistent trial 210

8.1 Screen layout for exploratory study 236

9.1 Example of an 'option' screen 244
9.2 User structure for E3 258
9.3 Example of User Model for Object first syntax group 260
9.4 Example of User Model for Action first syntax group 261
List of Tables

4.1 Overall time taken to complete tasks 105
4.2 Time taken to complete each task 106
4.3 Number of tasks attempted by subjects 107
4.4 Percentage of instances of object => action syntax (TopView subjects) 120

6.1 Overall number of errors by group 156
6.2 Number of errors of each type by group 159
6.3 Number of errors made on congruent and incongruent trials 160
6.4 Summary table of results (E1) 175

7.1 Number of syntax errors (S1-16) 194
7.2 Number of syntax errors (S17-32) 194
7.3 Number of sub-optimal responses (S1-16) 196
7.4 Number of sub-optimal responses (S17-32) 197
7.5 Combined errors and sub-optimal responses (S1-16) 198
7.6 Combined errors and sub-optimal responses (S17-32) 198
7.7 Mean time to select first command (S1-16) 200
7.8 Mean time to select first command (S17-32) 200
7.9 Summary table of results (E2) 211
7.10 General results table 212

9.1 Mean number of command correct on first attempt per task, for each group 250
9.2 Mean time taken for each task, by group 251
9.3 Mean number of commands used for each task, by group 252
9.4 Mean number of pop-ups accessed on each task, by group 253
Acknowledgements

I should like to express my thanks to the following for helping me in the completion of this thesis:

- My supervisor, Professor John Long, for his stimulating guidance, extensive feedback and all-round encouragement throughout the period of research.

- My secondary supervisor, Dr Andy Whitefield, for reviewing my work and providing much invaluable advice about doing a PhD.

- The members of the Ergonomics Unit for making the Unit such a friendly and stimulating place to work. Special thanks are due to my fellow PhD. students whose support and understanding has been greatly appreciated.

- The SERC and IBM (UK) Laboratories Ltd., Hursley, for their financial support in the form of a CASE studentship.

- The Human Factors Department at IBM for allowing me to make use of their facilities and expertise.

- My site supervisor, John Woolsey, for the enthusiasm and interest which he showed in my work.

- My subjects who willingly gave up their time to take part in my experiments.

Finally, I would like to thank my family and friends for their constant encouragement and support, especially Frank Wilson for keeping me going and providing me with a Mac at home.
Chapter 1
Introduction

1. Introduction

This chapter introduces the topic under investigation in the thesis and outlines the way in which the investigation will be conducted.

It begins by contextualizing the research within the discipline of cognitive ergonomics.

Then the chapter establishes that there are problems with the definition of an object-oriented interface and argues that providing a characterisation should form part of the investigation.

Object-oriented interfaces are shown to be a topic worthy of investigation because of their impact on the Personal Computing market and hence their introduction into the workplace, and because they are heralded as a development in easy-to-use interfaces.

Next, the form of the investigation is described. It is expected to involve different activities and here, their processes and products are outlined.

Finally, there is an overview of the organization of the subsequent chapters and brief details about their contribution.

2. The Thesis as a Contribution to Cognitive Ergonomics

Ergonomics sets out to optimize the relationship between people and their work by applying scientific knowledge to the design of tasks within the workplace. The introduction of the computer into the workplace brought a new challenge for ergonomics. Some of the problems associated with its use could be solved by traditional methods, but a set of problems emerged which could not be addressed using knowledge from the usual sources. These new problems were associated with the representation of the task entities. It was found that the representation held by the computer often differed from that held by the user and that mismatches lead users to form inaccurate expectations of the consequences of their actions and to be bemused by the actions of the computer. In order to address these problems, the new science of cognitive psychology was called upon and a new form of ergonomics emerged, cognitive ergonomics (see Long 1987 for a more complete description).
As a simple example of mismatch between representations, consider a word-processor in which any saved text is held in a file. Here, for the computer the characteristics of the particular text are unimportant, each piece of text occupies one file. However, the user may have different representations for pieces of text e.g. one may be a letter and another a report. For the user, the items have different functions and so s/he may expect that the difference should be indicated in the computer version of the task. In this case, the expectation is inappropriate and could lead to errors and confusion; for instance, the user may be inclined to specify the type of text instead of realising that the filename is sufficient.

This thesis hopes to investigate the topic of object-oriented interfaces from the perspective of cognitive ergonomics by comparing the representations of the task held by the computer and those held by the user. Particular attention will be paid to the combination of characteristics which either enhance or detract from the compatibility between the two representations. It is expected that the discovery of such information will be informative for the design of future user-computer interfaces.

3. The Topic Under Investigation: Object-oriented Interfaces

Innovations in computer system technology have led to the evolution of a novel class of user interface. These interfaces have come to be known as 'object-oriented interfaces'. Despite the market success and apparent popularity of these interfaces, very little research has been reported that bears upon the topic directly. This thesis aims to document the implications of object-oriented interfaces for the success of the interaction between the human and the computer. Towards this aim an investigation of the interface class will be conducted.

Here, the term 'interface' is used to refer to the components of both the computer and the user which subserve the interaction between them. So, on the computer side, it includes such components as the screen display and the input device while on the user side, it might include the knowledge about the meaning of command words and the ability to execute motor programs necessary for providing input. Further discussion about the nature of the interface appears in Chapter 3.

Commonly, user interfaces are classified according to certain features of the computer which are assumed to entail certain features of the user - thus the grouping is pertinent on both sides of the interface. For example, there is a class of command-line interfaces which require a command to be typed in response to a prompt, for interaction to occur. This style entails the user feature, a knowledge of legal commands should be present.
Clearly, the question of how object-oriented interfaces are distinguished from other classes of user interface is paramount to this investigation. Preliminary study revealed that there is no straightforward answer to this question. There is no agreed set of features which serves to distinguish object-oriented interfaces from other classes. Therefore, the investigation of object-oriented interfaces will have two complementary goals:

- to provide a characterisation of object-oriented interfaces
- to assess object-oriented interfaces with respect to their ease-of-use

3.1 What is an Object-oriented Interface?

Clearly to begin the investigation some common view of what might be called an object-oriented interface, is needed. This section aims to give a general picture, a 'flavour', of the sort of user-computer interface which is recognised as being object-oriented.

Use of the term 'object-oriented' has become increasingly widespread since the early 1980s. The term is readily applied as a descriptor and indeed there appears to be some consensus as to the type of computer system which it designates. However, no agreed definition exists and users of the term are unwilling to provide an explanation of its use.

As a starting point, the types of responses gained from asking users of the term what it means will be described.

In general, there are two types of response, one is based on the provision of examples while the other focusses on a particular interface feature. The first type of response is to give a set of examples of extant systems which can be described in this way. Systems which are cited include: Apple Lisa; Apple Macintosh; windowing software like Microsoft Windows, Gem (from Digital Research) and Taxi (from Epson); and the integrated package, Intuitive Solution (from Intuitive Systems). It is widely held that some systems are more object-oriented than others: implying that a continuum exists. Again, further probing fails to elicit explicit criteria against which such judgements can be made.

All of the systems mentioned above have features in common. However, it is not clear which of the features indicate the presence of object-orient edness. A problem of potentially confounding features arises as all the systems have a WIMP interface, support a Direct Manipulation interaction style and are based on a metaphor. These three potentially confounding features will be described briefly.
A WIMP interface is one that has Windows; Icons; Menus and a Pointer (although sometimes the M is said to stand for Mouse and the P for Pop-up menus), so it is concerned with a particular configuration of input and output methods. Since the technology became available, many interfaces have been implemented with this style, because it has been said that they are easy to use. However, it is not clear that they are easy to use in all circumstances - some tasks seem better suited to conventional styles (e.g. wordprocessing) and it has been reported that expert users often find the style to hinder rather than assist their work e.g. due to their familiarity with the computer, the visual prompting of commands is unnecessary. (See PC User, December 1986, for a statement of the negative aspects of WIMPs).

Direct manipulation, in the simplest terms means what it says: that one can manipulate things directly. With respect to the interface, it is implied that things which are to be manipulated have an obvious, visible representation and that the manipulation is performed on the object with the least number of arbitrary translations (compare dragging an icon across the screen by the movement of mouse <DIRECT> with moving it by typing in the co-ordinates of its required position <INDIRECT>). The term was introduced by Shneiderman in 1982 and since then it has undergone further analysis (Hutchins, Hollan & Norman 1986). Direct manipulation and its relationship with object-orientedness will be discussed at greater length in a later chapter.

Metaphor here refers to metaphor at the task domain level (as proposed by Hutchins 1986), i.e. where entities of a particular domain are represented in the computer version of the task. This use of metaphor is expected to increase the probability of intuitive use of a system by capitalizing on the user's existing knowledge. All the interfaces listed above are based (to some extent) on the desktop metaphor i.e. their entities are represented as items that might be found in a real-world office (e.g. folders; in-trays; clocks; documents etc.). Depending on the fidelity of a particular instantiation of a metaphor, the user can perform tasks by simply transporting known procedures to the computer representation - e.g. instead of typing in a 'delete' command to dispose of a document, the user simply 'puts' the representation of the document into the wastebin representation. Other metaphors are utilized in other application domains e.g. the control panel metaphor.

It is not clear what relationship exists between these characteristics and the concept of object-orientedness. Often careless use of terms implies that these characteristics can be equated with object-orientedness.

The second approach towards an explanation is to describe one prominent feature of such computer systems: the syntax of the command sequence.
In an object-oriented interface the command sequence is expressed as:

\[
\text{OBJECT} \quad \rightarrow \quad \text{ACTION} \\
\text{(noun)} \quad \rightarrow \quad \text{(verb)}
\]

e.g. Apple Lisa document \(\rightarrow\) duplicate

This ordering contrasts with the sequence found in traditional systems (either command-line or menu-based) which is:

\[
\text{ACTION} \quad \rightarrow \quad \text{OBJECT} \\
\text{(verb)} \quad \rightarrow \quad \text{(noun)}
\]

e.g. IBM PC-DOS delete \(\rightarrow\) filename

Although this approach to an explanation is an advance on the first approach - in that it makes some attempt at abstracting a general property which can be correlated with the presence of object-orientedness - it remains limited. It tends to imply that object-oriented interfaces can only be distinguished by the syntax of their command sequence, so reducing the concept of object-orientedness to a single factor. However, at the same time, users of the term are careful to stress that the reverse is true: there is more to object-orientedness than a particular syntax, but explicit descriptions to augment the concept are not forthcoming.

Examination of extant systems which have an object=>action syntax shows that this feature concurs with the characteristics described above (WIMP, Direct Manipulation etc.) Again, it is difficult to gain a clear understanding of the nature of an object-oriented interface in the presence of confounding features. If there were a good exemplar of an extant object-oriented interface which had object=>action syntax, but did not have a WIMP interface etc., comparison of it and the usual exemplars would yield information about their common object-oriented features.

There is a need then, for a characterization of object-oriented interfaces which clarifies their nature with respect to other interfaces.
3.2. Why Investigate Object-oriented Interfaces I?: The Emergence and Market Success of the Object-oriented Interface

To justify that the subject of object-oriented interfaces merits study in the field of cognitive ergonomics, it should be shown that this style of interface has achieved widespread use within the workplace. A brief history follows concerning the application of the term 'object-oriented interface' and the development of the interfaces it is used to describe.

Terminology in the domain of Human Computer Interaction (HCI) has to evolve to accommodate technological developments. In this decade, developments have occurred at a rapid rate and so terms have been hurriedly introduced and adopted without any rigorous specification being laid out for the boundaries of their application. Since use of terminology is influenced by technological developments, the developments which have led to the introduction of the term, 'object-oriented interface' should be examined.

In the 1970s, the Learning Research Group at the Xerox Palo Alto Research Center (PARC) developed Smalltalk. The Smalltalk system can be viewed in many ways. It is, at the same time, a programming language, a programming development environment and a windowing environment, amongst other things. For the first time, a system was seen that had windows, a direct manipulation interaction style and programs which were developed through the specification of objects. (See BYTE August 1981 for a series of articles describing various aspects of Smalltalk).

Following the novel developments seen in the Smalltalk system, in 1981, Xerox introduced the Xerox 8010 'Star' workstation. This office workstation represented a new departure in interface design. Unusually the development was driven by the requirements of the end-user interface. In other words, the conceptual model of how the user would relate to the system was determined first (by consideration of a number of principles from cognitive psychology) then the hardware and software were developed to accommodate this conceptual model.

The design process included explicit stages of prototyping and human factors testing to ensure that the best methods of implementation were adopted. Novel features of the final product included: a high resolution, bit-mapped graphics screen, icons (these are pictorial representations of task entities), a mouse input device for controlling cursor movement, a consistent syntax (selection of an object, followed by selection of the action), the integration of a number of media (text, graphics, tables etc), property sheets for specifying object properties, and a WYSIWYG (What-You-See-Is-What-You-Get) display. (See Smith, Irby, Kimball and Verplank, BYTE, April 1982 for further details of the Star user interface).
Following the lead of the Xerox Star, Apple launched the Lisa in 1983. This was a new development in the personal computer (PC) market. Here was an entire system: hardware, operating system, input device and software, all oriented towards a novel interaction philosophy. Many features which were introduced in the Xerox Star were incorporated including: windows; direct manipulation via a mouse; iconic representation of entities; integration of text and graphics environments etc. It was thought that the Lisa would prove to be an invaluable tool for professional office-based workers. In a survey of integrated software packages, Longley & Shain (1985), state that: "The Lisa will fit in with the way most managers work in that it is easy to keep an eye on several tasks at once and to skip between them, data transfer is simple and logical".

Though the Lisa was an acknowledged critical success, it did not succeed as a commercial product. Firms were unable to pay such a high price (around £7000) for what was essentially a one-user machine and so penetration into the PC market was low.

Recognising their strategic mistake, Apple were quick to launch a smaller and cheaper version of the Lisa. The Macintosh appeared in the UK in January 1984 and was heralded as "Lisa technology for everyone in a super pint-size package" (Apple User 1986).

The Macintosh has made an enormous impact on the PC market. Due to its popularity, it has undergone rapid development and a series of upgraded machines have been introduced. The market demands machines with larger memory capacity, faster processing speed and greater flexibility. Thus, in the last six years since its introduction, the Macintosh has appeared in at least nine guises, the main developments are listed below.

<table>
<thead>
<tr>
<th>Year</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1984</td>
<td>Macintosh introduced</td>
</tr>
<tr>
<td>1985</td>
<td>Macintosh 512 and HD20 Hard Disk Drive</td>
</tr>
<tr>
<td>1986</td>
<td>MacPlus</td>
</tr>
<tr>
<td>1988</td>
<td>Macintosh SE</td>
</tr>
<tr>
<td></td>
<td>Macintosh II</td>
</tr>
<tr>
<td></td>
<td>Macintosh IIx</td>
</tr>
<tr>
<td>1989</td>
<td>Macintosh IIcx</td>
</tr>
<tr>
<td></td>
<td>Macintosh IIci</td>
</tr>
<tr>
<td>1990</td>
<td>Macintosh IIfx</td>
</tr>
</tbody>
</table>

Accompanying these hardware upgrades, there have been many rapid developments in third party software for the Macintosh. Initially, the only software available for the Macintosh was that written by Apple: MacWrite (word-processing); MacDraw (drawing package); MacPaint (graphics package). Between 1984 and 1985, one hundred software companies agreed to write applications for the Macintosh (Longley and Shain, 1985).

The MacNews/Times 1000 Survey, a survey specially commissioned by MacNews (MacNews,
31 July 1989), demonstrates that the Macintosh is becoming an increasingly popular choice with corporate users. Information on Macintosh use was gathered from a random sample of companies from the Times 1000 list of the UK's leading banks, industrials and insurance companies. It was found that one in five large companies now have Macintoshes installed. Consequently, Apple's share of the corporate market has shown a dramatic improvement over the last five years; from 3% in 1984 to 19% in 1986 and to 27% in 1988. The survey also reports on reasons why Macintoshes are purchased and provides information about the range of tasks that they are used to support.

Figure 1.1 shows that the Macintosh is used to support a wide range of common office tasks - its use is certainly not restricted to graphical or desktop publishing tasks, although it is particularly renowned for supporting such activities because of its WYSIWYG facility and the way in which it capitalizes on spatial input.

![Figure 1.1](chart.png)

First, current and main Mac applications among corporate users (MacNews/Times 1000 Survey, MacNews, July 1989)
Figure 1.2 lends support to the notion that the popularity of the Macintosh is still strongly linked with its ease-of-use - this will be discussed further in the next section.

Following the introduction of the Macintosh, a proliferation of new packages were released which incorporated similar user interface features. Many of these packages (e.g. Microsoft Windows; Digital Research's Gem) were designed to run on the IBM PC (and later other popular PCs such as the Amstrad). They were often sold as 'operating environments' i.e. they were intended to replace the original command based operating system (PC DOS). This was achieved by representing files, directories and applications as icons; having pull-down menus for displaying commands/options and having windows for viewing more than one item at a time.

Clearly systems such as the Macintosh represent a successful development in the PC market. Companies and individuals are buying them in increasing quantity and the number of upgrades and new products demonstrate that this sort of technology is growing.

So, to what do they owe their success? One argument proposed is that this breed of computer systems has tapped a relatively new market - that of the non-computer professional.
As Information Technology becomes increasingly widespread in the workplace, a larger number of non-computer professionals i.e. those whose jobs are not directly focussed around computer use (e.g. accountants; teachers; doctors; receptionists), are required to use computers in their everyday work. This user population, which has been dubbed 'computer shy', is characterised as being unconfident with computers and possibly hostile towards their introduction.

The new style of interface is thought to win the approval of the inexperienced user for at least two reasons. Firstly, they 'protect' the user from the intricacies of the workings of the program - the user needs to know very little about the mechanisms of the operating system in order to perform effectively. The same was not true of earlier office systems where users would be required to type in large and apparently meaningless strings of commands and would, periodically, be confronted with daunting error messages. In fact, the difference between operating a computer (being an end-user) and programming one, did not seem to be very great. Secondly, these systems seem to have an inherent attraction which has been likened to that of video games. Both the graphical display and the reversibility of many functions entice the hostile user into exploring the system.

This section shows that the term object-oriented has been introduced to describe an important and (relatively) stable development in interface technology. It seems safe to predict that the development is here to stay, in the near future at least, therefore investigation is in order.

3.3 Why Investigate Object-oriented Interfaces II?: The Apparent Ease-of-use of Object-oriented Interfaces

A further reason for investigating object-oriented interfaces is their reputation for ease-of-use, which seems to underlie their popularity. This reputation has been enhanced by the advertising strategy for these products. In recent years a large number of claims have been made both in regular advertising and in review features in magazines such as BYTE, PC User etc..

Some representative quotes which help to perpetuate this reputation include:

Any program that can't be learned in a day or less is of little use to us. Fortunately nearly every program on the Macintosh can be mastered in that time.

[L. Page Maccubin, President of Page Bennett Associates, quoted in The Apple Family of Business Computers, Apple advertising material]
When the Lisa was first launched, Apple claimed that it would take a person only twenty minutes to learn the basic principles of how the system worked, and after that only another twenty minutes to become competent with each of the applications. This was compared with an Apple estimate of two days for learning how to use the average application package.


All studies of which we are aware have found that new users require less initial training, less subsequent training on new applications, and less ongoing support than IBM PC users.

[From the "Gartner Report", commissioned by Apple in 1987: reported in Apple Business, October 1988]

Windows is easy to use. Within 30 minutes I found most of the commands self-evident. The pull-down menus are convenient, and Windows programs in memory can be activated immediately with the mouse, from icons running along the lower border of the screen.

Gem Desktop is a treat to use and provides those who are shy of operating systems with a friendly alternative.

Yes - even the bugs in Taxi are user-friendly!

[extracts from a review of operating environments in PC User, December 1986]

Statements such as these suggest that this class of interface might be easier to use than other classes. If this were confirmed, then the investigation of object-oriented interfaces might be a fruitful way of discovering features which are responsible for user-computer interface usability, more generally. However, it has been mentioned previously that extant systems which are thought to have object-oriented interfaces also have confounding features (WIMP interfaces; direct manipulation; metaphor), so it needs to be determined whether the ease-of-use can be attributed to factors which are associated solely with object-orientedness, or whether it is a consequence of these other features. For example; is an interface easy to use because components of the task are represented as objects or is it because they are displayed pictorially, as icons?

4. The Investigative Approach

This section provides a brief outline of the approach of the work described in the thesis. Previous sections have established that:

i) there is little direct research on object-oriented interfaces in the human computer interaction literature;

ii) since their recent introduction, so called object-oriented interfaces have enjoyed great success in the PC market and their use is increasingly widespread in the workplace;
iii) An apparent reason for their popularity is their reputation for ease-of-use.

Given these justifications for studying object-oriented interfaces within the context of cognitive ergonomics, it is necessary to specify an appropriate approach for the undertaking. Since this interface class is still a relatively new development, the work will take the form of an investigation. Undoubtedly there is a need for some clarification, for example, to specify what an object-oriented interface is relative to common exemplars and to assess ease-of-use being careful to tease out which aspects are really due to object-orientenedness and which result from other confounding features (WIMP's etc.).

Towards providing this clarification, the investigation comprises two complementary activities, one analytic and one empirical. Each of these activities is associated with an output. The analytic activity focusses on setting up a framework which clarifies the topic area and provides a structure within which aspects of the interface can be modelled, while the empirical activity has as its output a set of experimental findings pertinent to the ease-of-use of object-oriented interfaces.

It is expected that the two activities will not be conducted independently, but instead they will be interleaved such that the framework is used to guide the direction of the experimental work, while, in turn, the results of the experimental work are used to modify and augment the framework.

Thus, the thesis hopes to contribute to the knowledge about object-oriented interfaces by providing a clarifying framework which can be used to characterise interfaces with respect to their object-orientenedness as well as a set of experimental findings concerning the ease-of-use of aspects of object-oriented interfaces, as determined by the framework.

5. The Organization of the Thesis

This section contains a brief overview of the contents of the subsequent chapters of the thesis and their contribution to the investigation.

The analytic activity begins in Chapter 2, where the requirements for a framework are discussed and a tentative framework is proposed. The framework includes a structure for modelling devices and users. In Chapter 3, a set of dimensions for the structure for modelling devices are derived from observation and from the literature. These dimensions enable devices to be modelled with respect to their degree of object-orientenedness.
Chapter 4 marks the onset of the empirical activity as it reports a comparative observational study. Here, two extant interfaces are modelled using the framework. User performance is then compared over the two interfaces. This study allows the relative ease-of-use of the two devices to be determined and provides information about user performance with respect to the dimensions used for modelling.

Following the observational study, a particular device feature is selected to be the focus for further experimentation. This feature is syntax (at the Communications or command-sequence level). The rationale for its selection, the proposed method for testing and a rudimentary user model are given in Chapter 5.

Chapters 6 and 7 report further empirical activity in the form of two experimental studies. The studies are designed to explore the effects of Input/Output level and Task level factors, respectively, on syntax usage. The results of each study are used to advance the user model with respect to syntax.

In Chapter 8, the results of the experimental studies and the observational study are discussed and a discrepancy is noted. Two views of the results are contrasted. Chapter 9 reports an experiment where use of alternative syntax sequences is observed under naturalistic conditions. The results of the study are discussed with respect to the two views. The results of this experiment are integrated into the user model.

The final chapter, Chapter 10, reviews the investigation and discusses a number of general issues in the light of the findings. Some limitations of the present work are identified and suggestions are made for ways in which the work could be extended.
Chapter 2
The Framework

1. Introduction

In Chapter 1, it was established that there is little research reported on object-oriented interfaces in the human computer interaction literature, despite their popularity and reputation for ease-of-use. It was proposed that the topic be investigated and that the investigation should encompass two sorts of activities, analytic activities and empirical activities.

This chapter reports the first stage of the analytic activity: the proposal of a framework for describing interfaces with respect to their object-orientatedness.

Initially, the need for a framework, within the context of this investigation, is outlined. The question of what is a framework is then considered. Once the general form and properties of frameworks have been established, the particular requirements for this framework are specified.

In response to the requirements, a framework is proposed. The framework comprises two components: its scope and its structure. The scope is further sub-divided into distinctions and definitions and it largely concerns clarification, while the structure enables the modelling of devices, users and the interactions between them. As framework development is expected to continue throughout the investigation, the framework proposed at this stage constitutes a 'first version'. It is expected that this will be refined as the investigation progresses.

The chapter concludes with a summary of its key points.

2. The Need for a Framework

As stated in Chapter 1, the output from the analytic activity will be a framework. The overall aim of the framework will be to make explicit the characteristics of an interface which determine whether it is object-oriented. Achieving this purpose will generate an understanding of what it is to be object-oriented (in interface terms) and will enable the modelling of particular interfaces with respect to their object-orientatedness.

Additionally, determining the nature of an object-oriented interface is a necessary precursor
to investigating its ease-of-use. Before an object-oriented interface can be evaluated as such, it is essential to be able to classify it as an object-oriented interface.

Preliminary investigation into the topic of object-oriented interfaces revealed that it is an area beset with confusion. Many of the terms which are associated with the concept (for example, the notion of an object) are ambiguous and imprecise. Frequently, such terms exist in common parlance, which leads to the mistaken belief that everyone knows what they mean and that their meaning is obvious and consistent. When it comes to describing interfaces, such assumptions are unhelpful and lead to misunderstandings. Each person works on the assumption that the common term has a common referent and so when s/he uses the term others will register the intended meaning, but too often this is not the case.

In order to counter such problems, the framework must play a dual role, fulfilling two purposes; providing explicit clarification, perhaps by specifying some definitions, as well as enabling interfaces to be modelled.

3. What is a framework?

In this section, the nature of frameworks in general will be considered, in order to determine that a framework is an appropriate structure for achieving the purposes described above.

The term, 'framework', itself suffers from the problems discussed above. It occurs frequently in everyday usage and suffers from being applied loosely. Thus, there is a risk that its meaning has become artificially broadened. In general usage, it is used to mean, "a structure into which completing parts can be fitted", so it can appear in almost any domain. Frequently, speakers offer a, "framework for discussion", where issues of interest are delineated. For the purposes here, it is necessary to reclaim the term, 'framework' from the everyday sphere and specify a more precise meaning.

To this end, Long's (1987) paper proposing a framework for user models will be considered. The framework was developed in order to clarify the confusion concerning user models. First, Long offers a definition of a framework. He goes on to state explicitly the scope of the particular framework of interest and illustrate its application.

The definition of a framework which is given in this paper reads:

A framework is an analytic structure which characterises the objects of its scope. By providing a common description for objects, a framework makes possible the establishment of identity, similarity and difference relations between them.
NOTE: henceforth, in the discussion of frameworks, the term 'entities' will be substituted for 'objects' with no difference in meaning intended.

The particular framework presented by Long has User Models as the entities of its scope. It has three parts:

i) Model Distinction
The framework differentiates between two classes of model: those that are User's models, i.e. possessed by the user and those that are User Models, i.e. models of the user.

ii) Model Definition
The framework provides a general definition of 'model' such that it is suitable for application to both the types of model distinguished above. It also sets out a more detailed description of the components of a model and the relationship between these and the entity modelled.

iii) Model Structure
The framework includes a model structure to characterise the genesis and purpose of models. This is:

An agent (A) creates a model (M) expressed as a representation (R) of an entity (E) for a user (U) having a purpose (P).

This structure serves to clarify the status of a model.

From the above, some general characteristics of frameworks can be deduced:

- frameworks are developed to disambiguate potentially confusing areas.
- the entities within the scope of a framework are made explicit.

Also, frameworks may incorporate:
- statements about important distinctions between its entities;
- useful definitions concerning the entities;
- a specification of a structure which might be applied to the entities in order to describe them in common terms.

In these terms, frameworks seem well suited to the task of clarifying ambiguous topic areas.
They achieve their purpose by two means:
i) providing clarifying statements e.g. distinctions and definitions;
ii) providing a structure for describing the entities of different areas in common terms.

The relationship between frameworks and models should be noted at this point. Frameworks describe a generic structure which can be used to express models and which is, thus, common to all the models it generates. By contrast, each model describes a particular instance of the framework. The framework, then, is used to instantiate particular models, and the models inherit a common structure by dint of the instantiation.

4. The Requirements for a Framework

As a framework would appear to be appropriate to support the two functions of clarification and modelling, the particular requirements of the framework for characterizing object-oriented interfaces should be specified, in terms similar to those listed above, as follows:

i) The entities of the scope of the framework
The framework should have all user computer interfaces as the entities of its scope, so that they can be judged as object-oriented or not on the basis of their common description.

ii) The components of the framework
It is clear that the framework should have two parts to serve the two purposes of clarification and modelling. The first part should be concerned with setting the scope of the framework by:
- making explicit important distinctions
- providing definitions of terms commonly used.

The second part should specify a general structure which enables particular instances of devices, users and interactions to be modelled. (It is expected that modelling devices will result in common descriptions to which criteria for object-orientedness can be applied).

5. Proposal for a Framework

In the following sections, the components of the framework will be proposed as specified in the previous section. The components are the scope and the structure, where the scope is further sub-divided into some distinctions and a set of definitions. The structure will be proposed in a general form and then specific versions for device modelling and user modelling will be distinguished. Little will be said about the content of the device structure: this will
be the focus of a subsequent chapter (Chapter 3).

5.1 Scope 1: Set of Distinctions

In this section, distinctions will be drawn between different aspects of the interface. It is apparent that a number of terms used to denote items within the interface are often used in different contexts, but that the nature of the context is left implicit. Consequently, confusion concerning referents of terms arises.

An example is the use of the word 'object': it can be used to refer to a physical object (such as a file in a particular office) or a computer representation of an object (such as the folder icon on the Apple Macintosh) or a user's cognitive representation of an object (such as the concept of a file). Often there is no indication of the intended context. Therefore, some division of the field of reference needs to be made explicit.

The framework proposes the following distinctions:

1. Let there be a Universe (U).

2. U is made up of three worlds:
   - the Real-World (RW)
   - the Device World (DW)
   - the User's Cognitive World (UW)

Each of these worlds is populated by objects, which can be described in terms of their features (the primitives of the Universe) - more details concerning objects and features are provided in Section 5.2.

In the following sections, each World will be described briefly and then the relationships which hold between them will be discussed. Note that all three worlds can be thought of as Researcher's views.

5.1.1 The Real-World (RW) and Domain of Application

The RW encompasses all the objects which might be included in any potential domain of application. In other words, the objects of a particular domain of application can be considered a sub-set of the RW. It is the subset of the RW, the domain of application, which
is of interest here as this 'world' will be influenced by the user-computer system. The user-computer system can influence this world by effecting some change of state in its objects. The change of state of an object or objects is achieved through the performance of a task. So here, a task comprises an interaction between the user and computer in order to bring about some change of state of the objects in the domain of application. Objects in the domain of application can be concrete (e.g. a ruler; a vase) or abstract (e.g. a design; knowledge about medical diagnosis).

As an illustration of the domain of application, consider the task of shopping using a mail-order, videotex system, the domain of application here constitutes the goods, as the user seeks their transfer them from seller to buyer in exchange for money. (see Buckley & Long, 1985 for more details of teleshopping)

5.1.2 The Device World (DW)

The DW refers to the structure associated with the device, i.e. the computer's processes and representation of the domain of application. For example, in a videotex system the computer displays textual and graphical descriptions of the goods for sale. These representations are the objects of the device world. While some of the objects and actions which exist in the DW are directly representational of the domain, the DW may also include objects and actions specific to the workings of the device, these might only be indirectly related to the representation of the domain e.g. the file structure in a word-processing system.

5.1.3 The User World (UW)

The UW, like the DW, refers to the structure associated with the user i.e. cognitive processes and representations both of the domain of application and of the DW. The nature of the representation is open to question. Debates have raged in cognitive psychology for many years concerning the way in which knowledge is represented. Candidate models, which have been considered, include propositional-based models such as semantic networks, schemas etc. and analogical models such as mental images (see Rumelhart & Norman 1985 for a discussion of knowledge representation). The debate is likely to continue with theoretical preference being motivated by the suitability of the representation for a particular purpose.

5.1.4 The Relationship between the Three Worlds

The three worlds are related in at least two ways. Firstly, they are related by the notion of a
As described above, a task can be said to have been performed when an interaction occurs between a user and a computer to bring about a change of state in the objects of the domain of application (a sub-set of the real-world). The interaction between the computer and user involves manipulation of the objects that exist in the DW and the UW to achieve some change of state of the objects in the domain of application.

For example, in sending a mail message the user carries out operations on the objects in the UW (e.g. the representation of the recipient; the representation of the information to be conveyed etc.) and manipulates the objects in the DW (e.g. the representation of the message; the representation of the recipient etc.) in order to bring about a change in the objects of the domain of application i.e. the recipient receives the intended information.

Secondly, the worlds have representational relations between them. This can best be illustrated by a diagram, see Figure 2.1.

In the diagram, the worlds are represented by boxes and the relationships by arrows. The domain of application exists as a sub-set of the real-world. The user has representations of objects which exist in the real-world but are beyond the scope of the domain of application, the device might also have such representations. Usually these representations are not called upon for the performance of a task, however, there are exceptions. For example, consider the 'delete' function in the Apple Macintosh. In the device world, the function is represented by an object, a 'wastebin' icon which looks like a dustbin. It is reasonable to
expect that the user has a mental representation for dustbins, but this would not be a representation of an object in the domain of application, but rather of a Real World object. In general, dustbins do not exist within the domain of office tasks, but in this instance they are used as a charicature of the wastebasket, the object which does exist as part of the domain of application.

RELATION A (Domain/UW) The UW includes representations of the objects of the domain of application. The nature of the representation is unknown though a number of different models have been suggested e.g. semantic networks (Quillian 1968; Collins and Quillian 1970); schemas (Norman, Rumelhart et al., 1975) frames (Minsky 1975); images (Kosslyn & Schwartz 1978) etc. The user may also have representations of objects outside a particular domain of application which exist in the Real World.

RELATION B (Domain/DW) The DW includes a representation of the objects of the domain of application. In the domain of office tasks; the DW would contain representations of objects such as files, folders, a calculator etc. The device might also have representations of objects in the real-world.

RELATION C (DW/UW) The DW may include a model of the user (this may be implicit and/or very rudimentary).

RELATION D (UW/DW) The UW includes a model or representation of the device world. The user forms concepts about the representations present in the DW - these may or may not be consistent with the user's representations of the corresponding objects in the domain of application (formed by relation A).

5.2 Scope 2: Definitions

This part of the framework supplies the definitions for important terms within the context of object-oriented interfaces.

The purpose of these definitions is to aid the accurate use of terminology. Each definition is given at a level that is sufficiently general to be applicable across each of the Worlds (RW; DW; UW) described above.

5.2.1 Features

As mentioned above, features are the primitives of the Universe - which means they cannot
be re-described in more basic terms. They can be of a number of types (from 1 to n). Two types have been identified as useful for defining and classifying objects. These two types are:

i) Physical features - features which have to do with physical structure typically expressed as appearance e.g. 'is rectangular'; 'is blue'; 'is solid' etc.

ii) Functional features - features which concern the behaviour of an object e.g. 'it moves up and down'; 'it contains liquid'; 'it squeaks' etc.

It is the perception of features that enables us to discriminate objects, one from another, and classify them into groups based on similarities.

As human beings, our senses are continually bombarded by a large number of stimuli. The resulting perceptions have to be organized to provide us with a meaningful experience. For this to happen, some grouping principles must be at work. Some grouping must occur even to recognize single entities as whole, complete and separate objects. How this happens is not to be discussed here, but it must be recognised that this process is responsible for our capability to distinguish objects on the basis of the jumble of sensations of the world.

Following the discrimination of objects as discrete entities, we apply a process of categorization whereby objects having similarities are classed together. Psychological research on categories and language suggests that there are universal, perceivable structures by which objects are grouped and that labels are assigned according to the perceived distinctions - thus while the labels themselves are arbitrary, the relations between them are not (e.g. Rosch and Mervis, 1975).

This process of classification could be motivated by economy of semantic memory - it is needless to store a large quantity of information about each particular object in a class, when much of that material is common across all instances.

Rosch et al. (1976) have proposed that objects are categorized at a number of levels, termed sub-ordinate; basic and superordinate. This levels refer to the hierarchical arrangement of many object categories. For example, a particular chair can be classed as FURNITURE (superordinate); CHAIR (basic) and PARKER KNOLL RECLINER (subordinate). It is at the basic level where the class boundaries are the most easily discriminable. It is also at this level that we most frequently refer to objects.

It appears that the level of classification depends upon which features are taken into account e.g. in the case of a dalmatian dog; its classification as an animal (superordinate) depends upon the consideration of features such as:
- is animate
- can move
- requires oxygen for life support
etc.

whereas inclusion in the dalmatian class (subordinate) requires different features to be considered:

- is white with black spots
- can retrieve items in mouth
- has a smooth coat
etc.

The two types of features described above (physical and functional) have been found sufficient for categorizing or labelling objects at the basic level. Knowledge about an object concept, e.g. chairs, is likely to include a generalized feature set for that concept, e.g. 'has a seat'; 'can support human body weight' etc. How that knowledge is represented and utilized is not at issue here. Assuming this is the case, for any particular object to be labelled, e.g. my office chair, a process of feature matching must occur in order to judge that it has sufficient features in common with the generalized feature set to allow categorization as an instance of 'chair'. It is conjectured that this process need only involve consideration of two feature types.

To examine the relative roles of the two feature types in the classification or labelling of objects (at the basic level) the possible combinations of the two feature types will be considered.

For any object concept there are four possible combinations, along the dimensions of appearance and behaviour:
For an object $x$ (e.g. a cup)

- **High**
  - Does not look like an $x$
  - Acts like an $x$
e.g. a cup without a handle; a tin can used for drinking
  - Looks like an $x$
  - Acts like an $x$
e.g. a cup

- **Low**
  - Does not look like an $x$
  - Does not act like an $x$
e.g. a knife
  - Looks like an $x$
  - Does not act like an $x$
e.g. a cup with a hole in the bottom

---

**Figure 2.2**

Classification of Objects with respect to Appearance and Behaviour

The four objects represented in Figure 2.2, a, b, c, d, are described below.

**a** represents objects that are peripheral to the class - such as broken exemplars, or where an object, normally considered to be from a different class, mimics the behaviour of this class. There could be some uncertainty about the application of the object term to members of this cell, at least without some qualification (e.g. a tin-can cup; a broken cup etc.);

**b** represents a stereotypical instance of an object - a good candidate for class inclusion;

**c** represents the case of 'it is not an $x$'. If it is true that something which neither acts nor looks like a member of a class is never a member of that class, then it appears that functional and physical features are sufficient for describing objects;

**d** represents objects which have an element of surprise such as 'joke' objects. A cup made from porous material would fit into this cell - it looks like a cup, but unfortunately it will not hold your tea!
It would seem from comparing a and d that possession of the functional features of an object type is more crucial to the definition of that object than possession of the physical features. This argument suggests that there is a weighting of feature type relative to applying an appropriate label.

From observation, it seems that a change in the features of an object may lead to a re-labelling of that object; either by supplying a completely new label (object has changed class) or by modifying the existing label with the addition of a qualifier ('a broken cup' etc.)

5.2.2 Object

Objects are the basic constituent of the worlds, consequently they are central to the description of object-oriented interfaces. Therefore, some clear concept of what constitutes an object is needed in order to ensure consistent usage.

'Object' is a term used frequently in everyday conversation, yet it is not clear that it is used consistently or in the same way. The ambiguity of the term can be attributed to its apparent simplicity i.e. 'of course, we all know what an object is'. However, for the purposes here, that is, the classification of interfaces, the term has to be more rigorously specified.

Definition:

An object can be described as the sum of the set of its features where features are the most primitive elements of the universe, i.e. they cannot be re-described in terms of more basic elements.

So an object \( O \) having four features of type 1 and five features of type 2 may be described as:

\[
O = t_1 \{ f_1 + f_2 + f_3 + f_4 \} + t_2 \{ f_1 + f_2 + f_3 + f_4 + f_5 \}
\]

In effect, it is impossible to specify a complete feature set for any given object in the Real World. Instead it is sufficient to rely on a partial specification in terms of the features pertinent to a purpose, here the classification.

The process of classification of objects (as mentioned in Section 5.2.1) can be accounted for under this scheme by the definition of object classes in terms of the features necessary for class inclusion. For example, if a particular class requires particular features (e.g. \( t_1f_2; t_2f_3; t_2f_7 \)
to be present and one feature to be absent (e.g. tlf3) then any object which can be matched to this template can be considered as a member of the class.

It is in the nature of objects to be amenable to transformation i.e. that the feature set can be altered. As the feature set changes so the object can be said to change state. Here, state can be shown to be a particular configuration in time of a feature set.

E.g. Object 1 in state v has features:

\[ t1(f1 + f2 + f3 + f4) + t2(f1 + f2 + f3 + f4 + f5) \]

after a transformation (t) Object 1 is in state w and now has features:

\[ t1(f1 + f2 + f3) + t2(f1 + f2 + f3 + f4 + f5) \]

The process of transformation is achieved by actions.

5.2.3 Actions

Actions are necessary for the transformation of an object from one state to another.

Definition:

An action is the process by which a change in the feature set of an object is brought about.

Actions can only be considered to have a state as in the sense, 'is activated' or 'is not activated'. Consider a particular car which is dirty: the feature 'is dirty' would be present in the feature set for this object. The invocation of the action 'wash' would transform this car (or its representation) by removing the feature 'is dirty' from its feature set. Clearly the action itself has undergone no change and is catalytic in nature. The history of an action could be documented, but only in terms of the instances of its activation or of the objects which it has transformed over a particular time period.

5.3 Structure for Modelling Interfaces

This component of the framework is intended to aid the modelling of user-interfaces. It provides a structure which can be applied to obtain models of particular users or devices. The
basic structure is the same for both components. However, to facilitate the modelling process some further properties have been added to the structure to give a device structure and a user structure, respectively.

5.3.1 The Basic Structure

As a starting point to developing the structure, a number of interfaces were examined some of which had been judged as object-oriented by domain experts. It was quickly found that a description at a single level would not be sufficient to exploit all aspects of the contrast. Therefore, it was proposed that a structure encompassing a number of levels of description would be most useful.

A number of models for describing interfaces in the literature include the notion of levels of description (e.g. Clarke 1986, Moran 1983). Moran's (1983) Command Language Grammar model was designed to provide a description of the user interface from the perspective of the user. The model has three levels of description:

i) the Conceptual Component which details the abstract concepts about the task the user wants to achieve and is subdivided into the Task level and the Semantic level;
ii) the Communication Component which concerns the way in which the dialogue is conducted and is sub-divided into a Syntactic level and an Interaction level;
iii) the Physical Component which concerns the way in which the user physically interacts with the computer and is further sub-divided into a Spatial Layout level and a Device level.

Each of these levels provides a further refinement of the level of detail concerning a particular operation. So, a task is first specified in general terms, at the Task level and then it is re-described at lower and lower levels each permitting the specification of more and more detailed information about the necessary interactions for the task.

While the general format of this levels structure is attractive, i.e. some notion of a Task component; a Communication or Dialogue component and a Physical component, certain aspects of the Command Language Grammar scheme make it unsuitable for wholesale adoption. First, it focuses on specifying the interface from a psychological (user) perspective - while in the present investigation, the emphasis is towards the device, since an extant style is to be explored. Second, Moran hints at the limitations of the levels i.e. the failure to capture 'syntactic' tasks and the absence of a 'conceptual model' at the Interaction level for describing the concept of the command line. Given the potential primacy of syntax for object-oriented interfaces (as mentioned in Chapter 1), it might be more useful to adopt a
structure in which there is potential for syntax (and semantics) to be specified at more than one level.

The three levels which are thought to be the most appropriate for the purpose are:

Task Level
Communications Level
Input/Output (I/O) Level

At each level the lexis can be specified and the syntax and semantics described. The Task Level has as its elements the sub-task e.g. the task of arranging a meeting is composed of a number of discriminable activities such as booking a venue; notifying participants; preparing an agenda; distributing the agenda; putting on refreshments etc., each activity can be viewed as a sub-task. The lexis of the Communications Level is the command word or its equivalent i.e. the means of communicating elements of the task, e.g. memo; document; send; create etc.. The lexis of the I/O Level is the unit in which input or output is communicated e.g. keystrokes; mousepresses; text characters; graphical elements etc.

5.3.2 The Device Structure

The device structure will specify additional information useful for modelling devices within the levels structure described above.

It has been ascertained by informal observation and discussion that object-orientedness is not a unitary characteristic i.e. it cannot be determined on the basis of a single interface characteristic or feature. In other words, there is no one characteristic whose presence can define an interface as object-oriented, so no straightforward rule exists of the type:

IF feature F is present THEN interface I is object-oriented

Instead, object-orientedness is thought to be multi-dimensional. For example, it might include an amalgam of different features such as a particular syntax; a certain manner of representation; specific types of task support etc. The device structure should, therefore, designate the dimensions which are relevant, thus enabling devices to be modelled in terms of common descriptions. A number of criteria can then be applied to the descriptions in order to judge how object-oriented a device is and in what respects.

Just as object-oriented interfaces elude a precise definition so does their contrast. It is not at all clear what sort of interface would fail to meet any of the object-oriented criteria. A
possibility is an action-oriented interface (the 'traditional' interface): though the meaning of term is not well specified. Here, the term 'non-object-oriented' will be adopted to refer to interfaces which are not classed as object-oriented. It is recognised that the term is not intended to convey any information as to the nature of the contrast set; other than its lack of object-orientedness.

The content of the dimensions of the device structure will be the subject of the next chapter.

5.3.3 The User Structure

Like the device structure, the user structure includes additional information to facilitate the production of user models within the framework levels structure.

Models of the user in the literature vary a great deal. It would not be appropriate to produce a psychologically real and complete user model - it would be far too time consuming and would result in much unnecessary information. So the nature of the user model depends upon the aspects of the user which are modelled and the purpose of the model (Whitefield 1987).

The form of model favoured here is one which specifies the processes performed and the representations held by the user during the performance of a task. It contrasts with models which focuss on the way in which knowledge is represented. This type of model has been chosen because the focus of the investigation is the usability of a device for the performance of a task, i.e. during an on-going interaction. The model will enable exploration of the match (or otherwise) between the device's representation of the task and the user's representation of the task. The stages of interaction required by the device (since most devices are relatively inflexible) can be viewed alongside the processes and representations of the user.

Users can be conceptualized in terms of a set of structures which support behaviour. The basic set of structures should be sufficient to support any instance of behaviour (see Dowell & Long, 1989). Here, a structural model of the user will be proposed which specifies the set of processes and representations held by the user. This general structure will enable production of models of user behaviour in the instance. For example, the structure may specify a representation, (words), in the instance this representation could have the value, (set, clock).

The user structure will be expressed in a form which has the following general properties:

i) that it can have any number of levels of description [L (1-n)];
ii) that it can have any number of representations at a particular level [R (1-n)];
iii) that it can have any number of representations at different levels [R (1-n)];
iv) that it can have any number of processes which operate to transform representations within or between levels \([P (1-n)]\).

The basic form of the user structure is shown in Figure 2.3. Here, a number of processes, \(P1 - P7\), act to transform one representation into another, \(R1 - R6\). Some of the processes transform representations between levels of description, e.g. \(P2\), while others operate within levels of description, e.g. \(P3\). At one level, all the processes can be thought of as processes which transform, however, in later versions of the structure the nature of the processes will be specified in greater detail. This structure is restricted to three levels of description (Task Level; communications Level; I/O Level), corresponding to those described in Section 5.3.1.

The structure shows two functions; the right hand side of the structure is concerned with the input or assimilation of task information while the left hand side is concerned with the output or implementation of the task. It is intended that for a particular instance of behaviour, a task instruction can be 'fed' into the right hand side of the structure and its transformation into a specification for user action will be shown in the successive stages from the right hand side of the structure to the left hand side.
Using this structure, user models can be formulated which reflect the results of the empirical activity. It is expected that each of these models will contribute to a more complete understanding of the user during interaction with devices with specified degrees of object-orientedness.

6. Concluding Remarks

The beginning of the analytic activity has been reported in this chapter. It has resulted in the proposal of a first version of a framework for characterizing object-oriented interfaces. The framework has two components; a scope and a structure, which serve to fulfil its two purposes of clarifying the topic area and providing a means for modelling interfaces. The next chapter will continue the analytic activity with specifying the contents of the device structure, i.e. the dimensions which are important for distinguishing interfaces.
Chapter 3
Deriving the Dimensions for the Device Structure

1. Introduction

This chapter continues with the analytic activity concentrating on the device structure, outlined in Chapter 2. It aims to specify the dimensions of the device structure i.e. the set of dimensions which are thought to be important for the classification of devices with respect to their object-orientedness.

First, the concept of a dimension is described in more detail and the mechanism for classification using the dimensions is outlined.

Then the process of deriving dimensions is begun. The set of dimensions are derived through the investigation of two information sources. These are:
i) observing extant interfaces - judged to be object-oriented by domain experts;
ii) reviewing the literature.

The nature of the interface to be described by these dimensions is considered. Finally, the results of the investigations are compared and integrated to form a putative set of dimensions for the device structure.

2. Dimensions and the Mechanism for Classification

It has been proposed that the device structure should embody a number of dimensions along which particular devices may vary, to facilitate the classification of devices. This section describes the notion of a dimension in more detail and explains how dimensions will be used in classification.

2.1 What is a Dimension?

A dimension represents a range of different features. These features must be related in that each one is a particular value of a common property. The dimension then represents all possible values of that property.

For example, some related features are 'is 5mm long'; 'is 10m long'; 'is 82 feet long' and 'is 4000 miles long' - the dimension in this instance represents all values of the property 'LENGTH'.
with each feature being a particular value along that dimension. In this example, the
dimension is continuous in nature, i.e. features can assume any value between its start and end
points. The start point is unclear in absolute terms, as it would be the shortest measurable
length (regardless of unit), similarly the end point would be the longest measurable length.
However, dimensions can also be discrete e.g. the dimension representing types of wood would
have distinct feature values e.g. 'pine'; 'mahogany'; 'chipboard'; 'sandalwood' etc. There
would be no particular start and end points unless some other notion were combined with it, to
form a different dimensions, such as 'softness of wood'.

Given a number of dimensions for describing interfaces, the feature values for particular
instances of devices can be stated, thus forming a common description.

2.2 How Dimensions Work in Classification

The device structure will include a number of dimensions (at whichever level of description is
most appropriate). Modelling a particular device will involve specifying its feature values
on each of these dimensions, resulting in a description of it in common terms. The dimensions
will each be associated with a criterion for specifying the value(s) which is necessary for a
device to be considered object-oriented in that respect (i.e. in terms of that dimension).
Application of these criteria will indicate to what extent and how that device is
object-oriented. For example, if the dimension represented the type of syntax of the command
sequence, the criterion might state that the necessary value for a device to be judged
object-oriented, is 'object first followed by action' ('object => action'). Any particular device
can then be classified in terms of that dimension, by determining its particular value on the
dimension, 'SYNTAX'.

The classification of devices can be represented pictorially, as shown in Figure 3.1:
The lines marked A, B, C, D, E represent the dimensions of the device structure. These dimensions range from values which are extremely non-object-oriented to those which are extremely object-oriented. This diagram shows three hypothetical devices (Y, O, X), represented in terms of their features on each of the dimensions. For each of the dimensions its criterion specifies the critical value at which something can be classified as object-oriented - these critical values are represented by the thick vertical line (this is shown as a straight line simply for convenience). Comparing each device representation against the critical values gives a measure of its object-orientedness. Here, device X would be considered non-object-oriented in terms of all the dimensions specified; device Y would be considered object-oriented in these respects and device O would be considered as mixed, i.e. it is object-oriented in some respects (dimensions B and D) but not in others (dimensions A, C and E).

This method enables interfaces to be classified qualitatively, but even this measure of precision is adequate as it affords a more precise view of what it is to be object-oriented than is generally found at present.
3. Observation of Extant Systems

One route to specifying what the dimensions might be, is to examine extant interfaces, both those that are thought to be object-oriented and those that are thought to be non-object-oriented. Comparison of the two types is likely to lead to some notions concerning the dimensions on which the two types of system differ, i.e. that the two types of system would reliably possess features of different values on the dimension.

To this end, an informal exercise was undertaken in which a number of extant interfaces were examined. These interfaces had been classified consistently by domain experts (e.g. human factors engineers specialising in the user interface) as tending to be object-oriented or non-object-oriented. Thus the approach taken was to examine instances to abstract possible general dimensions for classification. This approach was consistent with the exploratory nature of the exercise.

From the observations, a number of possible dimensions have been identified. These have been discussed with the domain experts in order to gain a more complete picture by drawing from their knowledge of other (perhaps not yet commercial) interfaces.

Some of the interfaces which were included in the observations were:

**OBJECT-ORIENTED:**
Apple Lisa; Apple Macintosh; Digital Research's GEM; Xerox Viewpoint; Metaphor from Metaphor Computing Systems (documentation only).

**NON-OBJECT-ORIENTED:**
Lotus Development Corporation's Lotus 1-2-3; IBM TopView; Dbase; Ashton-Tate's Framework.

A number of potential dimensions were identified and these are described and discussed in the following sections.

3.1 The Representation of Task Elements

The object-oriented interfaces all had the striking feature that the user is initially presented with a set of object representations. These are generally representations of objects which are a part of the non-computer supported versions of the task e.g. document, folder, filing cabinet etc., although they can represent novel objects which are specific to the device supported
version of a task e.g. 'directory' (of files).

In the observed systems these objects were represented iconically, examples are shown in Figure 3.2:

![Icons of objects](image)

- **out-tray** (from Xerox Star)
- **envelope pad** (from Intuitive Solution)
- **folder** (from Gem)

**Figure 3.2**
Examples of object icons

However, experts agree that the objects can also be represented textually:

- graph 2.86
- note
- in-tray

The textually presented objects have only been noted in experimental systems.

Some of the extant systems allow alternative representations e.g. the Macintosh offers a textual representation where the items are ordered chronologically, alphabetically or by size. However, these textual representations function differently from the iconic ones. The user can no longer move the items around, placing them in any position on the screen, their position is determined by the structure governing the list.

For the purposes here, the two types of representation will be considered equivalent, in that they yield two sorts of information:

i) Information about an object's presence and its particular identity. However, the two representations may differ in the amount of information which they convey in this respect. Often icons convey more information about an object's features, either by depicting them in some way or because they imply more strongly that the representational object has features in common with the real-world object. Some icons even provide information about changes in the features of the object represented e.g. in the Apple Macintosh the wastebin icon alters its
image when the wastebin becomes full, the two images are shown in Figure 3.3.

![Wastebin - empty](image1) ![Wastebin - full](image2)

Figure 3.3
Two images of the wastebin (from Apple Macintosh)

ii) Information about the object's location.
In many extant iconic interfaces the user can determine the position of the representational objects - by dragging the icons around the screen - this ensures that the user is aware of locational information. It is not essential that the position of icons be user determined for location information to be conveyed, provided that particular objects are seen to have a consistent and discriminable position on the screen. This organization contrasts with a list arrangement where items are placed according to the listing principle and have no position in their own right.

It is clear that the use of icons per se has implications for the usability of an interface. Indeed, research has been conducted concerning many aspects of icon usage: how they are comprehended and how their meaning is retained (Hemenway 1982); the distinctiveness of their visual features (Arend et al. 1987); their utility in aiding recall memory (Lansdale 1988); an assessment of their usefulness in supporting human computer interaction (Rogers 1989) etc.

The link between iconic interfaces and the representation of objects in the interface is strong. Possibly this has arisen because it has been found that icons are better suited to the representation of objects than actions. It is difficult to design concrete icons for actions; some solutions which have been tried include:

- using some sterotypical object e.g. 'scissors' for 'cutting'; 'post-box' for 'sending'
- using a number of arrows to indicate the action, as shown in Figure 3.4.
Figure 3.4
Example of an action icon, using an arrow - displaying a start state and an end state of some object, as shown in Figure 3.5.

Figure 3.5
Example of an action icon which shows two states of an object

In general such strategies have not been found satisfactory, perhaps because the motion or change in time associated with most actions cannot be displayed easily without exploiting some form of animation.

A strong trend has grown up for treating object-oriented and iconic systems as synonymous (Cox & Hunt 1986), but it seems misguided. The use of icons in an interface has implications at the level of the device's representation of the task. The important issues concern the degree to which the icons are successful in conveying the appropriate information about the device's objects and operations. On the other hand, the orientation of an interface towards objects seems to have implications for the way in which the device organises the task. Clearly, these two characteristics of interfaces need to be distinguished.
So how is this dimension manifest in non-object-oriented interfaces? How would non-object-oriented interfaces be characterised with respect to the representation of task elements? There seem to be at least two ways in which they might contrast with object-oriented interfaces:

i) by initially presenting the user with a display of action representations;
ii) by representing objects in a different way e.g. as items in lists which have no unique position.

Both alternatives have been identified, to some extent, in extant non-object-oriented interfaces.

Feasibly, the presentation of available actions could be achieved iconically as in Figures 3.4 and 3.5 or textually:

e.g.

```
edit
copy
delete
```

The initial presentation of actions on the screen has been observed in an experimental system, where a number of actions were displayed as menu items. However, in many extant systems, the distinction is not so clear. The user is presented with a mixture of object and action terms, often in menu format e.g. in Lotus 1-2-3 the top-level menu gives the user the selection:

```
Worksheet Range Copy Move File Print Graph Data Quit
```

Alternatively, a command-line style interface simply displays a blank line with some type of prompt (e.g. > or ? etc.) where an action word is implicitly expected. An example of this type of interfaces is the 'dot prompt' option in dBASE II. The '.' can be followed by any of the large selection of command words e.g. BROWSE, SUM, CREATE etc. The command words generally indicate actions.

No interface has been found of the command-line style which expects an object word to be typed in first. The objects in these interfaces are generally all represented as one object type, files (though there may be different classes of files e.g. text files; executable files etc.). The files can be addressed via their filename.
As well as having actions as their 'starting point' for interaction, the non-object-oriented interfaces have a subtly different manner of representation for objects. Often objects appear in the guise of files and directories, with different object classes corresponding to the different file types. At first sight this may not seem very different from a text representation of objects. However, the object names are simply a list. A particular list entry does not strongly imply the identity of a particular object and no location information is available. The object's position is determined by the listing principle.

3.2 Limitation of Command Set

The second feature that is particularly noticeable in the object-oriented interfaces is that the set of available actions for each object is limited to those which are valid for that object. The validity of a particular action being dependent on the features of an object and its particular state.

In practice, this limitation is usually manifest in the presentation of action options in a menu (usually a pop-up menu). So to perform a task, the user initially selects an object and then s/he can select the appropriate action to be performed on that object, from a limited set of actions. A menu is shown in Figure 3.6.

```
<table>
<thead>
<tr>
<th>Undo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cut</td>
</tr>
<tr>
<td>Paste</td>
</tr>
<tr>
<td>Duplicate</td>
</tr>
<tr>
<td>Round Corners ...</td>
</tr>
<tr>
<td>Smooth</td>
</tr>
<tr>
<td>Show Clipboard</td>
</tr>
</tbody>
</table>
```

Figure 3.6
An example of a limited option set, presented as a menu
This context limited presentation of options reduces the potential number of errors that a user can make, i.e. no errors of an illegal pairing of an object and an action can be made nor can an inappropriate action be performed on an object, e.g. 'format' would not be an available option for the object, 'wastebin'.

The notion of limitation could also be implemented in a command-line style of interface, but here the user could make illegal pairing errors, although probably the illegal function would be inhibited such that no unexpected side effects occurred, e.g. the user could type in 'delete clock', but the change implied by this command string might not take place and instead an error message would result.

The advantage of this kind of constraint mostly resides in the explicit presentation of a limited set. However, it is unclear whether this aids the user simply by reducing the potential for error or whether there is a more consistent mapping with the user's view of the world, i.e. that the user already holds some notion that objects are associated with a constrained action set.

There are two ways in which the non-object-oriented systems could differ in this respect. These are:

i) by not including any notion of a constrained response set;
ii) by having an opposite constrained response set i.e. a limited set of objects upon which a particular action can act.

In general, the non-object-oriented systems observed differed as in i) above. However, an experimental interface was observed which went some way towards ii). Here, actions were presented to the user and on selection of an action, a prompt box appeared which provided guidance as to the set of permissible arguments (objects). Obviously, this arrangement could be extended such that a list of permissible objects, given the current system state, was presented.

However, it might be that this structure, limiting the options to those that are legal, should be regarded as object-oriented, irrespective of the particular items that are presented within it (objects or actions). In other words, it does not matter whether the user is given a set of objects followed by a limited set of actions or the reverse - the very fact that there are options signals that there are objects (at a higher level of description) present.

For example, it is suggested that each of these two dialogues may be equivalently object-oriented, on the grounds of structure:
While this possibility of equivalence should be borne in mind, it is obvious that any effects peculiar to the items having specific values (e.g. greater representational compatibility with the user's representation of task goals, when objects are presented first) would reduce the utility of considering the two arrangements as equivalent.

3.3 Syntax of the Command Sequence

The feature which is most often identified as indicating an object-oriented interface is its command syntax. Object-oriented interfaces have the command sequence:

\[
\text{object} \Rightarrow \text{action}
\]

i.e. the object is selected first (usually from the selection present on the screen - see 3.1) and then an appropriate action is selected (usually from a limited option set - see 3.2). This arrangement of the command sequence is particularly noticeable because it is contrary to the expected natural English syntax for the imperative e.g. "Eat your dinner!"; "Find the elevator" etc.

This command syntax contrasts directly with the syntax found in most other types of interface which is:

\[
\text{action} \Rightarrow \text{object}
\]

Sometimes, the distinction between the two alternative command sequences becomes blurred. One system observed required the user to select a file from a list presented on the screen, using the cursor keys. Then, the name of a command to be applied to that file is typed in. So far,
this interface would seem to have an object => action syntax, as well as an initial representation of objects. However, in contrast to these features, the typed command word appears in front of the filename, thus mimicking an action => object command sequence.

e.g. the command word precedes the filename to which it is applied:-

    list of filenames:

    file1
    file2
    save letter
    memo1
    plan
    xxxxx

The ambiguity would be reduced if syntax were taken to refer to the order in which entities are entered, irrespective of their order of display.

3.4 Ratio between the Objects and Actions

A further possible factor for inclusion is the ratio between the number of objects (or more precisely object types) and the number of actions. Some object-oriented interfaces "object-ize" some of their functions. Thus, the number of objects is enhanced.

For example, instead of representing the function 'delete' as an action, the Apple Macintosh presents a wastebin icon which serves as a container for unwanted items (at the level of documents, folders etc. rather than at the alphanumeric character level). Here, deleting a document is accomplished by moving the icon representing that document into the wastebin icon. Similar instances are found in the Xerox Star interface, e.g. instead of the action, 'print', there is a representation of a printer. Printing is achieved by moving the document icon to the printer icon. A number of classification schemes for icons distinguish between icons to represent objects and icons to represent functions (see Gittins 1986, Jervell and Olson 1985)

Representing functions in object terms affects the ratio between the objects and actions, as one action is replaced by one object, the number of objects increases relative to the number of actions. It is surmised that object-oriented interfaces would tend to have a greater number of objects relative to actions, while non-object-oriented interfaces would display the opposite characteristic, of having a greater ratio of actions to objects.
Whether this dimension is robust as an indicator of object-oriented interfaces remains to be seen. It is almost certainly related to the presence of direct manipulation, the nature of this relationship will be discussed in a later chapter (see Chapter 10).

3.5 The Representation of Properties of Objects

Some object-oriented interfaces (e.g. Xerox Star) have an explicit representation of object properties. The representation reflects the hierarchical arrangement of the objects in that, for each object class, there is a template for properties and their possible values. Within a class each particular object can vary with respect to those properties within the range specified e.g. a particular text character may have the property BOLD face, where the class of text characters may have any one of five alternative face properties (e.g. italic, underline etc.).

Non-object-oriented interfaces differ in this respect as the properties of objects tend to be either implicit or not particularly associated with the object itself e.g. face characteristics of text are often assigned to text characters by indicating a portion of text then applying a function (e.g. in the VIEW word processor system). This implementation style tends to imply that a function is operating, here one of 'bold-ing'.

3.6 Concealing the Mechanics from the User

By virtue of their manner of object implementation, object-oriented interfaces are successful in shielding the user from what is going on "behind the scenes". If a user wants to work on a text document and then wishes to switch to drawing a graph, then s/he should not have to endure some lengthy, seemingly meaningless procedure. In some object-oriented interfaces (e.g. Apple Lisa; Apple Macintosh with Multifinder etc.) this is possible, all environments being fully integrated. The user does not have to complete work on one item before beginning work on another item (even if it is in a different medium).

For example, in Apple Lisa, a user may choose to edit a document which has been prepared under the LisaWrite application, s/he clicks on the document icon and it opens (into a window representation), ready for editing. After a few minutes, s/he may decide to look at a graph, (prepared under LisaGraph), in order to check some figures. The graph is opened in the same way by clicking on its icon. The user can view both items on the screen at once and can work on either item simply by clicking the mouse in the appropriate window. At no point does the
user have to specify or even be aware of the different applications which are supporting the word-processing activity and the graphical activity.

Contrastingly, many non-object-oriented interfaces are arranged so that the user explicitly has to change programs to work in a different medium.

This feature is related to the use of windows, as they are an enabling mechanism for allowing different tasks to be represented concurrently on screen. It is difficult to envisage how working on a number of tasks could be represented in the absence of windows.

### 3.7 Summary

The informal observation proved useful in providing a number of specific ideas for possible dimensions. Dimensions can be identified when the two sorts of system exhibit contrasting features e.g. they have a different syntax at the Communications Level. In addition, information has been gleaned concerning the criterion which might be set for each dimension to determine whether an interface should be classed as object-oriented.

### 4. Reviewing the Literature

The second approach to deriving dimensions is to review the literature. As mentioned previously, there is very little research that specifically concerns object-oriented interfaces, so a literature review based only on direct references to object-oriented interfaces would yield scant information about potential dimensions for classifying interfaces. Therefore, it is likely to be more fruitful to take a wider view of the literature, i.e. to consider references to the term, 'object-oriented' more generally. This approach should enable a general view of 'what it is to be object-oriented' to be abstracted. It is hoped that any such general notions can then be re-expressed in order to describe distinguishing interface characteristics.

The quest is unlikely to be straightforward as Robson (1981) said, 'the words "object-oriented" mean different things to different people', and this becomes evident.

The references will be described and discussed in the four following sections. These correspond with the main areas of application of the term, which are:

i) a class of programming languages;
ii) a class of systems;
iii) direct references to the end-user interface;
iv) miscellaneous references.

It should be noted that articles on both object-oriented programming and object-oriented systems are usually written from a computer science perspective and often the term, 'interface', designates the interface to the programmer/software engineer rather than to the end-user. Providing that one is alert to the potential confusion, this research remains valuable for the present purpose. It is likely to provide information concerning object-orientedness in general and there may be parallels to be drawn between the two types of interface from these different perspectives.

4.1 Object-oriented Programming

The term object-oriented programming was coined to describe the programming paradigm developed at Xerox PARC. This programming style was exhibited commercially in the Smalltalk-80 programming environment. Since its inception, many other languages have been developed which adopt its principles. These have been seen homogenous enough to form a category of programming languages (i.e. a class in contrast to declarative languages, procedural languages etc.). Examples of object-oriented languages are:

- Smalltalk-80
- Objective C
- Object-oriented Forth

A factor which probably helps to confound the application of the term object-oriented is the fact that the Xerox Star interface (a graphical, WIMP type office workstation) was developed around the same time and the term, object-oriented, has been applied in a liberal and unprincipled fashion to both the Star interface and the Smalltalk environment.

In general, object-oriented languages contrast with say, procedurally-oriented languages, because they are organised entirely in terms of objects. All the information required for the computer to perform the required tasks is contained within objects. In a programming sense, objects are defined as "a package of information and descriptions of its manipulations" (Robson 1981). All the elements of a problem which can be conceptualized as an object will then be defined by the programmer by specifying their attributes and the manipulations to which they are subjected. Each object is an instance of a class where the class provides all the necessary information to construct and use objects of its own kind. The manipulation of any object is achieved by sending a message to that object where a message is a specification of one
of the legal manipulations for that particular object. These manipulations may be described as a set of methods. On receipt of a message, the object indicates which of the methods should be invoked. Pascoe (1986) illustrates this mechanism using his own notation.

For example, an expression to take the square root of \( x \) will look like this:

\[
x : \sqrt{\cdot}
\]

Here, the object, integer \( x \), is being asked to perform a square root operation on itself; \( x \) is the receiver of the message :\sqrt{\cdot}. The manner in which this operation is to be executed is described within the object.

This message passing method of controlling events contrasts with that found in procedural languages, where events are specified more or less sequentially i.e. the programme is read and executed in order unless particular jumps are specified (e.g. via sub-routines, GOTO statements etc.).

It is often stated that object-oriented methods of programming reduce the complexity of programming tasks by helping the programmer to think in larger chunks. Object-oriented languages (and system development techniques) make it possible for the programmer to model closely the actual requirements of an application. For instance, if the programmer were building a system for aircraft flight control, the objects defined might include aeroplanes and airports. The object-oriented technique capitalizes on the programmer's knowledge about the real-world counterparts of the entities pertinent to an application by insisting that this be translated directly into the object's definition. (Guilfoyle, the Guardian 14th August 1986, Booch 1986). Object-oriented techniques contrast with more traditional approaches where the trend is to express everything in terms of procedures.

There is a certain amount of controversy concerning which features are necessary or defining for an object-oriented language. Nierstrasz (1988), offers a fairly loose definition, "any programming language that provides mechanisms that can be used to exploit encapsulation is (at least to some degree) object-oriented". Here, encapsulation refers to the decomposition of a system into smaller units, in an object-oriented language. This is achieved by expressing information in terms of objects. Nierstrasz supports the idea that languages can be more or less object-oriented, depending on the extent to which they exhibit encapsulation, the degree of homogeneity of the object model etc.

Pascoe (1986) adopted a more rigorous definition for object-oriented languages, requiring that all of the following four features be present:
- information hiding
- data abstraction
- dynamic binding
- inheritance.

Each of these will be described briefly.

Information hiding is a property which reduces the interdependence between modules of a programme. It is achieved by having the state of a software module contained in private variables which can only be accessed from within the scope of that module. The data in a module should only be manipulable by a set of localized procedures, so changes to one module should have few consequences for other modules.

Data abstraction is related to information hiding. It refers to the facility by which a programmer may define an abstract data type consisting of an internal representation, plus a set of procedures used to access and manipulate the data.

Dynamic binding is an enabling mechanism concerned with the way in which the specific details of how an operation on a particular data type is to be performed, are specified e.g. consider the operation, PRINT. It is performed in slightly different ways for integers and character strings. In procedure-oriented languages a case statement is included to check that the correct printing procedure used for an element's data type. However, in object-oriented languages the responsibility for having the appropriate printing procedure lies with the objects. The same message is sent to each object, but the way in which it is interpreted is dependent upon the information stored with the object (this is known as polymorphism).

Inheritance is a property that enables programmers to create new classes of objects by specifying the differences between a new class and an existing class (in terms of their features and behaviours) instead of starting from scratch each time. E.g. a new object, TRUMPET, may be created which is a specialization of the class MUSICAL-INSTRUMENTS. The TRUMPET would inherit behaviour that is appropriate for both BRASS-INSTRUMENTS and MUSICAL-INSTRUMENTS as well as having its own specific behaviours.

Whether a loose or a rigorous definition of object-oriented programming languages is adopted, a common aspect seems to emerge. The essential features are that the programme is divided up into chunks expressed as objects (which may correspond well with the objects found in the real-world problem domain i.e. the applications domain of the system); these objects are self-contained in that they specify their own procedures and the objects exist within a
hierarchical arrangement, in that each object is a member of a class (there may be a number of levels of classes), where class inclusion is determined on the basis of properties, and inheritance mechanisms may operate between super-ordinate and sub-ordinated classes.

4.2 Object-oriented Systems

The term, 'object-oriented system', is often found in the computer science literature and it is used to refer to systems which have their roots in object-oriented programming. However, the nature of the relationship between these systems and the object-oriented languages is unclear (it is uncertain whether object-oriented systems just closely mirror the features of the object-oriented programming languages or if they have to be implemented in an object-oriented language. However, that is not of particular concern here).

Robson (1981) contrasts object-oriented software systems with data/procedure-oriented software. He defines a software system as an information-manipulation tool in which the manipulation is described by software. The traditional, procedure-oriented software systems are said to be composed of a collection of data which represents some information and a set of procedures that manipulates the data. Changes occur in the system by invoking a procedure and providing some data for it to manipulate.

Robson believes object-oriented systems to differ from the procedure-oriented systems as they have a single type of entity, the object. It can be manipulated, like data, and yet it describes manipulations as well, like procedures. Information is manipulated by sending a message. The message specifies which manipulation the object is to perform on itself. He goes on to describe two properties of object-oriented systems. One is the distinction between classes and instances. The class describes an object or set of related objects (generic descriptor), whereas an instance is an actual object described by a particular class. The other is the inheritance mechanism whereby one object class can inherit the attributes of another object class. This mechanism is usually involved when one class is modified to create a new class.

A recent issue of the ACM Transactions on Office Information Systems (1988) was dedicated to the subject of object-oriented systems and in the Editorial to this special issue, Lochovsky defined object-oriented systems as having these characteristics:

- abstraction of data
- inheritance of properties
- encapsulation of data and operations
- persistency of data.
He provides a more stringent (though not conflicting) view of object-oriented systems than Robson. Unfortunately, these terms are left undefined. However he goes on to say that, "object-oriented systems appear to be a natural basis for modelling office data and activities. Offices consist of many entities (people, documents etc.) that usually interact with each other in fairly well-defined ways. Communication can be considered to be via message passing either verbally or by documents. Knowledge is partitioned among the entities in the office. Office entities perform their activities concurrently and asynchronously. All of these preceding characteristics seem to fit well with the object paradigm".

So it would seem that the important characteristics of object-oriented systems have much in common with those discussed in relation to programming languages e.g. objects being instances of classes; inheritance of properties; data and its manipulations being defined together within an object etc.

4.3 Some Direct References to Object-oriented Interfaces

There are a small number of direct references to object-oriented interfaces. Though these do not attempt to provide a definition, they give some further clues as to their nature. These references are discussed below.

Owen (1986), in an article on Direct Manipulation says, "one dominant characteristic of most frequently cited examples of Direct Manipulation is that they attempt to encourage users in the illusion that they are engaging with a clearly differentiated collection of objects, with varying degrees of analogy invited to the physical world. This approach defines a class of object-oriented interfaces which occupies one substantial corner of the Direct Manipulation spectrum". It would seem that Owen distinguishes the two concepts, "object-oriented" and "Direct Manipulation", though no explicit definitions are given. It would be interesting to find out what sorts of interface occupy the object-oriented corner of the class of Direct Manipulation interfaces, as compared with those that are simply Direct Manipulation interfaces. Such a comparison would be helpful in deconfounding the two concepts.

Continuing with his description of object-oriented Direct Manipulation interfaces he says that, "within the class there is considerable variation in the ways in which these illusions may be fostered, the extent to which the analogy with the physical world is pushed, the means provided for interaction with the objects, and the extent to which the underlying software reflects the approach. The definition is in principle independent of implementation language although the debt to object-oriented languages is clear".
Clearly, Owen believes an object-oriented interface to be something other than simply an interface developed in an object-oriented language. It is unclear both whether the illusion is just at a physical/representational level and how it would be manifest at any other. Also, it would seem that object-orientedness necessarily entails Direct Manipulation, as the class of object-oriented interfaces is described as a corner of the Direct Manipulation space. He does not make it clear whether there are other classes/possibilities of object-oriented interfaces which do not have this relationship or whether this is the only class of object-oriented interfaces.

Verplank (1988) writes along much the same lines when he describes a class of object-oriented direct manipulation interfaces, "which rely on concrete and visible objects, simplified sets of user actions and rapid feedback where the key activity is visibly moving screen images by pointing at them". He implies that this interface class is more narrowly determined than the general class of direct manipulation interfaces, though no further definitions or examples are given.

Object-oriented interfaces and direct manipulation techniques are linked again by Nierstrasz (1988). It is noted that "direct manipulation interfaces provide the user with the illusion that the objects of the application are being 'directly' manipulated by a set of polymorphic operators: move, copy, delete, resize, etc.". It is implied that direct manipulation is a most useful medium for interacting with an object-oriented interface structure. "Objects . . . suggest a corresponding interface paradigm: for every software object in the system we wish to represent or interact with, there can be a corresponding 'presentation object' that will be visible to the user on a display", (Nierstrasz and Tsichritzis 1988). Interaction then occurs via direct manipulation of these presentation objects.

As noted earlier, Cox and Hunt (1986) appear to equate object-oriented interfaces with iconic interfaces, when they say, "making computers easier to use has been an enduring dream since the dawn of computing. This is one of the reasons for the current interest in iconic or object-oriented user interfaces - interfaces that present information as pictures, instead of text and numbers." Clearly their view is that object-oriented interfaces are synonymous with iconic interfaces and while this is not necessarily contradictory to the view that object-oriented interfaces involve an 'object concept', independent of mode of display presentation (i.e. text vs. icons) it would seem to be just a particular instance which happens to have a graphical display. If, however, graphics are believed to be essential for object-orientedness, then this view would appear to be in opposition to the one derived so far.
4.4 Miscellaneous References to the Term, 'Object-oriented'

In this section, a number of miscellaneous references to object-oriented will be described. The first is a useful survey of object-oriented concepts which clearly is of major interest. The remaining references are more varied and are possibly of more peripheral interest.

In his survey of object-oriented concepts, Nierstrasz (1988), considers object-oriented languages and systems to determine whether being 'object-oriented' means the same in all applications. He determines that all approaches which are labelled object-oriented have one feature in common - their ability to support encapsulation, i.e. the packaging of information into self-contained objects. He proposes that all other concepts generally regarded as being associated with the object-oriented approach (e.g. data hiding, inheritance, polymorphism etc.) ultimately depend on object encapsulation. Although, there is no explicit consideration of object-oriented interfaces, Nierstrasz does imply that he believes the notion of encapsulation extends to them.

The term 'object-oriented' is used to describe a task structure in a paper by Jorgensen and Barnard (1986). They report an experiment comparing two task structure styles. They discuss a task where four objects have to be processed by each of two operations. Such a task may be entering and validating four data items. This task can be attempted in two contrasting ways.

1] Each object is entered and then validated.
   
   Sequence:
   
   Object 1 -> enter -> validate
   Object 2 -> enter -> validate
   Object 3 -> enter -> validate
   Object 4 -> enter -> validate

   Here, each object is processed entirely before the next object is involved. This is termed an object-oriented structure.

2] The 'enter' operation is carried out first on all objects, followed by the 'validate' operation.
Sequence:

\[
\begin{align*}
( \text{object 1} ) & \quad \text{enter} & \quad ( \text{object 2} ) & \quad \text{then} & \quad \rightarrow & \quad \text{validate} & \quad ( \text{object 2} ) \\
( \text{object 3} ) & \quad ( \text{object 4} ) & \quad ( \text{object 3} ) & \quad ( \text{object 4} ) 
\end{align*}
\]

Here, each operation is finished before the next is begun. This is termed an operation-object structure.

The two styles were compared over performance on a message-handling system, it was revealed that the object-oriented task structure was found easier to learn in terms of time, errors and requests for help information.

In their experimental design, the command syntax and the task structure vary together. However, the syntax need not be determined by or be consistent with the task structure e.g. the operation-oriented structure could have an object => action syntax (object 1 - enter; object 2 - enter; object 3 - enter etc.). It may be interesting to conduct a further study varying task structure (object-oriented vs. operation-oriented) against syntax (object => action vs. action => object). In any case, this distinction between task structure and syntax should always be noted.

The use of the term 'object-oriented' here seems to be justified by the ordering principle at work i.e. the 'object-oriented' structure specifies the objects first and the fact that each object is processed as an independent entity during performance of the overall task.

A further interesting reference to 'object-oriented' is found in a paper by Horak and Kronert (1984) concerning standardised formats for documents. In the paper, they propose an object-oriented document architecture model. A standardised model is thought to be important for processing and interchanging documents between different systems.

The model presumes a document to have two independent, but related structures: a logical structure (the sectioning/organisation which underlies the document's comprehensibility e.g. abstract; table of contents; sections etc.) and a layout structure (the positioning of the material, such that its 2-D form appropriately reflects the logical structure).

It is proposed that each of the structures are hierarchical and are composed of objects. For each structure, there is a basic level of object, higher levels of the hierarchy containing composite objects made up from these basic objects.
As well as the hierarchical structure for each composite object, the organizational structure among its constituents must be specified. Objects are not only characterized by their constituents or content portions but also by their properties. Properties have both a type and a value. Relations may exist between objects and in which case these should be specified by type and value.

Objects are also said to be instances of object types. These types can be defined by name and different sets of rules.

Horak and Kronert depict their model as shown in Figure 3.8.
Figure 3.8
Document Architecture Model (Horak & Kronert)
Although the model outlined here is one which provides a comprehensive means of specifying a common format for any document, and is therefore of a static form, it is interesting that it is described as an object-oriented structure. Its main elements seem to be that its entities are described as objects which can be arranged hierarchically, such that composite objects of the various levels are made up from basic objects. Objects are instances of object types and each object can be characterised by its properties. Presumably, these are the features that qualify the model to be classed as object-oriented. The reliance upon objects and the type of relationships which are seen to hold between them, seem to parallel features found in other 'object-oriented' entities (e.g. programming languages, software systems).

4.5 Summary

Although the literature surveyed here covers a number of widely different applications of the term, 'object-oriented', it is clear that the referents of the term have a number of common elements.

Object-oriented seems to involve the packaging of information or data in terms of objects, where the objects are seen as being discrete entities, describable in terms of their properties, which in turn, determine their inclusion in a class of objects. A number of the references mention that inheritance mechanisms operate between classes and sub-classes of objects, such that objects at a lower level of specification inherit the properties from the parent objects. In programming, this arrangement is important for reducing the amount of code required for each new object. The literature on programming hints at a connection between an object and its allowable manipulations, indicating that data concerning an object's manipulations are stored with the object. As the manipulations refer to a particular object, it is implied that they are valid for that object. Presumably an object would not store information concerning manipulations that were not relevant.

The information gathered here will be integrated with that gained from the informal observations in the following section.

5. The Set of Derived Dimensions

This section brings together the information gathered in Sections 3 and 4, in order to specify a tentative set of dimensions.

First, the notion of 'interface' will be discussed to enable the dimensions to be specified appropriately. Two possible approaches towards considering the object-orientedness of the
interface will be outlined. Finally, the putative dimensions suggested by the two information sources will be integrated to form the device structure.

5.1 The Nature of the Interface and Ways to Specify its Object-orientedness.

The aim of this section is to affirm the nature of the interface for the purposes here. This exercise will ensure that the dimensions derived to describe the device aspects of the interface are appropriate.

Defining the interface is not a straightforward endeavour, as Card, Moran and Newell (1983) state, "the human-computer interface is easy to find in a gross way - just follow a data path outward from the computer's central processor until you stumble across a human being. Identifying its boundaries is a little more subtle."

As noted in the first chapter, an approximate scope for the interface is adequate for the purposes here. It states that the interface comprises those elements of the device and those elements of user which subserve the interaction between them. These elements can be concerned with either the presentation or reception of information necessary for communication.

For illustration, imagine that the interface in question is part of an office support system, designed to help with the production, modification and storage of documents. Here, the elements of the device and user which are part of the interface include; the device's representation of task entities (e.g. folders, documents, clock etc.) and the command words typed in by the user (e.g. copy doc.x, delete doc.y etc.). By contrast, elements which lie outside of the interface might be the user's knowledge of gardening and the device's toggle switch which does not impact on the interaction.

There are two alternative approaches to designating the device side of an interface as object-oriented and it is appropriate to consider them. The two approaches are:

i) an object-oriented interface is defined as one which has emerged from the use of object-oriented programming languages/paradigm.

ii) an object-oriented interface is seen as one which fits within the mould of object-orientedness as found in other contexts.

Some of the literature implies that i) is the case, many authors speak of object-oriented interfaces as being the product of applying the object-oriented paradigm. Under this view,
object-oriented interfaces are only considered to be object-oriented by virtue of their development, and no other characteristics are cited for recognising them as object-oriented.

However, literature concerning the relationship between programming languages (and classes of programming languages) and the type of problems which they solve, suggests that theoretically the language and the nature of the solution produced are independent. Though, in practice, some languages might be more convenient for providing certain types of solution e.g. SNOBOL was developed to deal with problems involving string manipulation and FORTRAN is particularly suited to problems involving numeric calculations which can be expressed in ordinary algebraic terms.

With regard to the relationship between object-oriented languages and object-oriented interfaces, it is thought that there is no problem in implementing user interfaces in SMALLTALK which are not object-oriented. So if non-object-oriented interfaces can be programmed in Smalltalk, then presumably object-oriented interfaces can be developed in languages other than Smalltalk (or any other object-oriented languages). Therefore, while there might be particular relationships between object-oriented interfaces and object-oriented languages, it is not the case that an object-oriented necessarily entails an object-oriented interface.

Having disputed the utility of the first approach, the second approach will be examined. It assumes that there are recognisable properties of being object-oriented. As described in Chapter 2, the approach taken here is to identify those properties in order to specify a device space within which interfaces can be modelled. In accordance with the view of interface given above, the likely dimensions upon which features responsible for object-orientedness lie, will refer to elements of the device which are associated with the interaction between device and user.

The following section lists the dimensions identified as a result of observing interfaces and reviewing the literature on object-oriented things.

5.2 Integrated Set of Possible Dimensions

Although the two information sources yielded different sorts of information, there was considerable overlap in terms of the general picture they conveyed. The observation approach led to the direct identification of a number of interface features which reliably differ between the two types of interface. Also, this approach provided information concerning possible criteria for determining what should be considered object-oriented. As the
information gained from the literature review is related to the dimensions suggested by the observations, then the dimensions will be presented as they were in Section 3, but information from the literature will be used to augment their description.

D1. Representation of task elements

This dimension concerns the way in which the information, necessary for performing tasks, is presented to the user.

Criterion = objects presented initially on the screen

A characteristic common to all the referents of the term object-oriented, in the literature, was that they organise their information in terms of objects. Objects are employed as conceptual building blocks (of either existing or novel content) from which a working knowledge of the environment can be built. This concept is common to programming, systems and the document architecture description.

The question could be raised as to whether this arrangement has any degree of useful similarity between instances (the nature of the objects in each case being quite different) or whether 'object' is so much of a general term that it forfeits its utility. The more abstract objects become, the less likely it is that they can be recognised, intuitively, as objects. However, in terms of the device aspects of the interface, the presentation of information in terms of objects is distinguishable from its contrast (i.e. presentation in terms of actions).

D2. Limitation of command set

This dimension concerns the limitation of the available options to those that are valid.

Criterion = command set limited to the actions valid for a particular object, given its properties and current state

The literature hinted at objects being associated with some notion of constraint. Generally, objects entail some mention of allowable manipulations (the set of potential manipulations are limited to the set that are permissible given the nature of the object). A similar scheme is found in the Real World where knowledge about objects includes a sense of what actions are legal/possible e.g. our knowledge about pens, includes a realisation that pens can (generally) be picked up; can be used as a tool for writing; can be used as a stirrer for drinks (though it is recognised that this may interfere with future writing properties); but it is inappropriate to use pens as a container or for sticking up wallpaper.
D3. Syntax

This dimension concerns the ordering of entities, principally at the Communications level (i.e. command sequence syntax). Syntax can be object-oriented at the Task level if the task is structured appropriately, most devices are open with respect to task structure, i.e. it is the user who imposes the structure.

Criterion = command syntax should be object => action

The literature on object-oriented programming implied that an object => action syntax was in operation. A message is sent to an object and in Pascoe's example the object is specified first. Goldberg and Ross (1981) confirm this ordering when they describe teaching Smalltalk programming in schools. Here, an instance, 'jill', of the category, 'Box', is created via the instruction, 'jill <- Box new' and the box is made larger via 'jill grow: 50'.

D4. Ratio between Objects and Actions

This dimension concerns the ratio between the number of object types (no object instances) and the number of actions which are represented by the device.

Criterion = greater ratio of objects to actions

The literature offered no specific support for this dimension. However, the finding that object-orientedness is associated with expressing information in terms of objects is consistent with the idea of their being a higher object to action ratio.

D5. The representation of properties of objects

This dimension concerns the way in which object properties are represented.

Criterion = direct representation of an object's properties which is closely linked to the object representation.

In the literature, objects properties are part of an object's definition. Objects are described as being instances of classes where class inclusion is determined by properties. Properties are usually specified for a particular class and they are inherited by any instances of that class or any instances of its sub-classes.
D6. Concealing the mechanics of the device from the user

This dimension concerns the extent to which the user is shielded from the mechanics of the programmes of the device.

Criterion = the user should be less aware that s/he is interacting with a computer system than s/he is of working on the task in hand.

There is no direct support for this dimension in the literature - except that it supports the idea that information is presented in an object-centered manner, i.e. it helps to foster the illusion that the user is interacting with objects, as distinct from processes.

6. Summary

In this chapter, the analytic activity has proceeded to specify a tentative set of dimensions and associated criteria for the device space. It is expected that this set of dimensions will be successful in enabling the distinction of object-oriented interfaces from non-object-oriented interfaces. The next step is to validate the dimensions by showing them to have consequences for the user during the course of an interaction and to determine whether they are related to the ease-of-use of an interface.
Chapter 4
Validating the Dimensions: an Observational Study

1. Introduction

In this chapter, the first stage of the empirical activity is reported where the dimensions proposed in Chapter 3, are validated as having consequences for the user. The validation is achieved by comparing user performance over two devices which contrast in terms of the dimensions.

First, a domain of application is selected for observation and the characteristics of its associated user population are discussed.

Then, two contrasting devices are selected and modelled in terms of the dimensions of the space of possible devices. A number of expectations are proposed which predict in a general way how the performances of the two subject groups will differ. These expectations are based on the premise that object-oriented interfaces are easy to use.

The design, method and results of the study are reported and the results are discussed with respect to the expectations.

The chapter concludes with a consideration of the possible future directions for experimentation suggested by the findings.

2. Rationale for Observational Study and its Objectives

In Chapter 3, a number of tentative dimensions and criteria were proposed for classifying devices with respect to their object-orientedness. All of these dimensions focus on aspects of the device. Since this investigation focusses on the object-orientedness of the user interface, it is important to determine whether the types of distinctions made possible by the dimensions have consequences (positive or negative) for user performance.

Towards this end, a comparative observational study is proposed in which user performance is compared between two devices which vary with respect to their object-orientedness in terms of the dimensions. If any reliable differences are found in the performance of the two groups then it is presumed that they can be attributed to the different features on the
dimensions possessed by the devices.

Subject performance will be compared for carrying out a set of tasks using two devices which differ with respect to the dimensions. As well as gross measures of performance e.g. time taken; tasks completed etc., the subjects' errors and difficulties will be recorded. Comparison of these should enable insights to be made concerning the effects of the critical features of the device for the user.

In addition, the nature of the interfaces selected for this study allow a particular comparison to be carried out, that of determining which syntax sequence, object => action or action => object, is used by the subjects.

The study has two objectives:

i) a general objective - to determine whether different device features on the dimensions have consequences for user performance and to find out the nature of those consequences (i.e. facilitative or detractive) with a view to future experimentation.

ii) a specific objective - to discover whether subjects prefer to use a particular value on the dimension, syntax at the Communications Level.

A set of expectations have been generated which state that, in general, for each dimension, performance will be facilitated by the device having the object-oriented feature. These expectations are listed in Section 10.

3. Selection of Domain

The selection of an appropriate domain was guided by the availability of suitable extant computer systems. It was essential that the two systems chosen for comparison should differ significantly in their perceived degree of object-orientedness. Since most systems on the market which exhibit features of object-orientedness are office support systems (e.g. Xerox Star, Apple Macintosh, Intuitive Solution etc.) the domain of common office tasks was thought appropriate.

The introduction of Personal Computer workstations into the workplace has influenced the distribution of tasks. At one time professional workers (e.g. doctors, lawyers, advertising executives, personnel officers etc.) relied on a considerable amount of secretarial support in order to produce reports, letters, set up meetings etc. These work practices are now changing and professional workers are beginning to execute more of their own work, for example,
word-processing environments have enabled the non-typing professional to write up his/her own material and electronic communication systems allow arrangements to be made and material to be transferred from one's desk.

It seems that, whatever their particular field of expertise, professional workers require similar sorts of support— they will need to perform tasks such as writing reports (with text and graphics); communicating with colleagues; keeping an appointment diary; doing calculations etc. The domain of common office tasks includes all such activities.

A number of studies have reported on the work of managers (e.g. Mintzberg 1973; Kotter 1982), and they provide further evidence that there is considerable similarity between the sorts of task undertaken by executives, irrespective of their company or field. Mintzberg's analysis suggests that 'managers' jobs are remarkably alike, "the work of foremen, presidents, government administrators, and other managers can be described in terms of ten basic roles and six sets of working characteristics". The roles he describes include leader, disseminator, resource allocator and entrepreneur. Both studies draw attention to the fact that managers spend a great deal of their time communicating.

4. Selection of User Population

The target user population is part of the new market (mentioned in Chapter 1); professionals who want to use computers to support their everyday tasks, but who do not have any specific training for using computers. These people have expertise in a particular area (e.g. education, law, finance) and they are largely responsible for directing their work and organising their own time schedule. Therefore, it is expected that they are a reasonably intelligent group, but that they have no particular knowledge of computing or any strong desire to learn more than is necessary to execute their tasks. It is probable that their use of computers will be intermittent and that their main focus of attention will be on the content of their work (i.e. the details of a particular legal case; the accounts for a particular period etc.) rather than on the means of carrying it out.

The subjects selected for this study were expected to be representative of the user population described here. These subjects are described in Section 8.

5. Selection of Computer Systems

5.1 General
The two contrasting computer systems which have been chosen for this study are:

**Apple Lisa** v. 3.0B (with the applications, LisaWrite; LisaGraph; LisaList; LisaCalc).
Hereafter referred to as 'Lisa'.

**IBM TopView** 1.0 supporting IBM Personal Editor running on an IBM PC-XT (PC DOS v. 2).
Hereafter referred to as 'TopView'.

**NOTE:** from here on the particular configuration of software and hardware which make up the computer side of the interface will be referred to as the DEVICE.

Both of these devices support document production and allow some organization of information held on-line. Text only was used for two reasons:

i) equivalence - if only the basic word processing functions are used the two devices can support the same tasks and there is a minimal difference in terms of representation on the screen i.e. documents have the same layout and the text only differs by font and colour (black for Lisa; green for TopView). If graphical or drawing packages had been used, an equal degree of similarity would be difficult to achieve due to the differences in the functions of the packages and the resolution of the screen.

ii) minimalization of training - it was thought best to keep the training to a minimum. If a greater number of environments had been used a correspondingly longer training would have been required with little foreseeable gain in terms of the aims of this study. It is not the focus of this study to be concerned with the intricacies of interaction within a medium, e.g. editing a letter; drawing a graph etc. - but rather with a general interaction with the devices and the effects of their different styles.

### 5.2 Details about the Devices

There follows a brief description of each device. The features which are thought to be most important for this study are emphasized.

#### 5.2.1 Lisa

The Lisa was released in 1982. As mentioned in Chapter 1, the Lisa was modelled on the ideas of the Xerox 'Star' interface; being a smaller, less functional and cheaper version. It was an unusual product in the PC market because it was offered as a whole system and was originally sold as a package including hardware and software. Later components were offered...
separately. However, Lisa seemed to not find its place in the market (price vs functionality etc.) and was withdrawn from sale. It was superceded by the Macintosh which has reduced functionality, but which costs less.

The Lisa is a WIMP system and it (to some extent) supports the desktop metaphor as it represents various office entities iconically (e.g. wastebin, folders, documents). Its windows can overlap completely, giving the impression that one is working with separate pieces of paper etc. which can be stacked on top of each other. Lisa supports the Direct Manipulation method of interaction - the user can interact with task entities by manipulating their representation on the screen e.g. by moving icons. Interaction is achieved via the mouse input device. The mouse controls the movement of the pointer on the screen. It has one button and offers three functions:

i) selection;
ii) dragging, i.e. a techniques for moving items on screen;
iii) selection from a menu.

The user can also interact using the keyboard e.g. for text input.

NOTE: An accelerator feature is provided whereby objects can be accessed by doubleclicking on their icon. Doubleclicking was discouraged in the study and it was not described in the training session. This was because doubleclicking makes it more difficult to determine the actions of the subject by the screen view. Also, it is uncertain where doubleclicking stands with respect to command syntax at the Communications Level as it compresses selection of the object and selection of the action 'open' into one movement.

Figure 4.1 shows the components of the screen layout of the Lisa.
5.2.2 TopView

TopView is a product which was launched by IBM to support multi-tasking on the PC. It provides a front-end to PC-DOS (the PC disk operating system), i.e. protects the user from using DOS directly and instead presents a version of it. It also supports a number of application packages (the user chooses which s/he wants to use with it) and switching between these is possible without having to un-load the previous application. This means
that the user can use a word processing package for generating a document and then can switch to the simplified DOS in order to say, copy it and then return to the word processor without having to re-load it.

Additionally, TopView provides facilities for use by the applications it supports; the intention being to improve consistency. These facilities are a mouse, windows, and the TopView menu. However, whether these facilities can, in fact, be utilized depends very much on the application package which is used. Unfortunately, there is considerable variation amongst recommended packages as to their degree of compatibility with TopView. As TopView can be used with a number of independent products, there is no guarantee that these will be consistent (in terms of the details of interaction) with each other or with the TopView version of DOS.

TopView is a text-based system. It can be used with a mouse for selecting options and signalling the cursor position. It can support windows, but only with a few packages (Personal Editor cannot be used with windows).

TopView makes a distinction between the Pointer (which is used in TopView mode) and the Cursor (which is for text mode). However, when a mouse is used, this distinction becomes minimal as the pointer becomes a cursor when used in the appropriate areas.

The mouse in TopView is used for moving the pointer around the screen, it also has these three functions:

i) selection of items;
ii) signalling termination of an operation;
iii) accessing the TopView menu, a panel of nine options.

TopView's initial panel ('Start-A-Program') presents options for adding/changing/deleting the programs that run with TopView. It also provides a menu for choosing any of the applications that are set up to run with TopView and TopView's version of PC-DOS, 'DOS Services'. In this study, the subjects had a choice of DOS Services (for file management) and Personal Editor (for word processing). Both are described below.

**DOS Services**

DOS is usually command based - the user is presented with a prompt and s/he can issue any of a large number of commands which have many possible parameters. Examples of commands are:

```
TREE B:/F>TREE.LST
```
Issuing commands in DOS can be complicated as the user is required to remember complex formats and file addresses. In providing a front-end, TopView hopes to reduce the complexity while retaining a basic level of functionality (the most common operations). It aims to provide more visual guides and feedback to the user - the command words and available files are listed on panels and after an operation has been performed a message appears to inform the user. In DOS, success is only indicated by a reappearance of the prompt.

TopView DOS Services retains the hierarchical tree structuring of files of traditional DOS. The top directory is known as the root directory and this can contain files (of various kinds - text, programmes etc.) and subdirectories. Likewise, the subdirectories can contain both files and (sub-)subdirectories, so a tree structure is formed.

The address for a particular file is given by the path that must be traced through the subdirectories to reach that file.
E.g. the address of "Ref-GH" might be C:\TopView\Current\Ref-GH.

**Personal Editor**
The Personal Editor is a basic word processing environment. It initially presents the user with a panel where the path name of the file to be edited must be entered (in the correct position). The Personal Editor proper can then be entered. The Personal Editor screen has two areas: at the bottom a command line and above it the text area. To access a file the command, 'Edit <filename>' must be entered on the command line. The user moves from the command line to the text area by pressing the Escape key. Below the command line, system messages appear - a reminder of the current file name and any notification of errors.
The Personal Editor uses function keys and three were used in this study:

- F2 = Save
- F3 = (Save & ) File
- F4 = Quit

These were all labelled.

NOTE: The mouse becomes inoperable with the Personal Editor, except for accessing the TopView menu - which is of limited use.

Figure 4.2 shows the screen layout for TopView
Figure 4.2
Illustration of the TopView screen layout
5.3 Classification of the Two Devices in Terms of the Device Structure

The two devices are extremely dissimilar and only the pertinent differences will be described here, namely those which are thought to be important for object-orientatedness. Below each of the devices is considered with respect to the six dimensions and associated criteria set out in Chapter 3.

5.3.1 Representation of Entities.

This dimension concerns the way in which the information necessary to perform a task is represented. Object-oriented interfaces initially present the user with a set of objects. The actual style of representation can be either textual or iconic.

**LISA**

The Lisa initially presents the user with a selection of objects (e.g. wastebin; hard disk; folders; stationery pads etc.). The Lisa as used in this study used iconic representations for these objects. Each icon had a label and so the user has both identity and location information available. Other representations are possible such as lists.

The Lisa can be considered object-oriented in this respect as its initial display is of objects, where the representations give the impression of being independent entities.

**TOPVIEW**

TopView presents the user with a set of action commands in a menu panel. Another panel on the screen displays the names of files and directories. The display here is dependent on the current directory. The files and directories appear in a list and the position of an entry depends upon the current ordering constraint (e.g. items can be displayed by date, by size etc.), so the user does not have direct location information available. TopView only has textual representation of its entities. The objects and actions are distinguished by their location in separate screen areas.

TopView is fairly mixed with respect to representation of task entities as both actions and objects are presented initially. However, the objects represented seem to have the form of labels for file, rather than being independent objects.

5.3.2 Constraints on the Command Set

This dimension concerns the extent to which a command set is limited to those options which
are valid. In object-oriented interfaces the command set is limited to the actions which are valid for a particular object.

**LISA**
The action set for each object is constrained to those actions which are valid for the particular object type and the state of the object at a particular moment, e.g. if a document is in the state "open", then an 'open' option is not available from the menu.

The Lisa is object-oriented in terms of this dimension.

**TOPVIEW**
In TopView the given action set (a sub-group of DOS commands) is common across the objects. In some sense, TopView could be considered object-oriented in this respect, because the user is limited to these actions which are permissible for all objects. However, this becomes less true if the option to enter other DOS commands is chosen.

Lisa and TopView do not differ very greatly on this dimension.

### 5.3.3 Syntax (at the Communications Level)

This dimension concerns the required ordering of the command sequence, where object-oriented interfaces have an object => action syntax.

**LISA**
The required syntax is object => action. Syntax is much more difficult to specify for doubleclicking and direct manipulation, although for the latter, specify object; perform action (e.g. move), do still apply.

Lisa is clearly object-oriented on this dimension.

**TOPVIEW**
The required syntax for TopView is action => object. This ordering is suggested by:

- screen layout (actions appear on left hand side)
- cursor movement (cursor always returns to the action panel)
- tutorial (only action => object syntax is demonstrated)

However, the opposite syntax, object => action can be used to equivalent effect.

In standard use, TopView would be classified as non-object-oriented on this dimension,
however there is potential for interaction to occur in an object-oriented style if the user so wishes. (This alternative facility is capitalized upon, in that it is used as a metric in subsequent data capture).

5.3.4 Ratio of Objects to Actions.

This dimension concerns the ratio between the number of actions and the number of object types. It is suggested that object-oriented interfaces have a higher object:action ratio. The ratio is likely to be a result of the expression of many of their functions as objects.

**LISA**
The Lisa has a fairly large number of object types. Unfortunately, for the sake of equivalence and simplicity only three were exploited in this study: folders; text documents; wastebin. The disk object was also mentioned and was used by some subjects.

Lisa can be judged as relatively object-oriented on this factor.

**TOPVIEW**
TopView has two object types: files (in this case text files) and directories. All other functions are served by commands.

TopView would be classed as non-object-oriented on this dimension.

5.3.5 Representation of Object Properties

This dimension concerns the extent to which object properties are associated with a particular object and whether they lead the user to believe that there is a hierarchical arrangement of objects.

**LISA**
The Lisa has no obvious representation of object properties that is directly linked to objects. Properties of text objects (Font, Style etc.) are represented by ticks beside menu options. Other properties are represented visually e.g. folders can be seen to be containers because their windows can contain other icons etc.

In respect of this dimension, the Lisa is not particularly object-oriented.
TopView has no obvious representation of object properties. For each file, there is a display of its size, type and date of last modification, but due to the nature of its content the display is static i.e. not manipulable by the user.

TopView is not object-oriented in this respect.

5.3.6 Concealment of the Mechanics of the System

This dimension concerns the extent to which the workings of the device which are not strictly related to the task, are hidden from the user. In object-oriented interfaces, the user is more aware of working with the objects, than of interacting with programs.

Lisa

Any programmes required are automatically invoked. If a text document is accessed, then the editing/word processing environment is immediately available. The user does not have to signal the need for different applications. Furthermore, when two different sorts of application are required (e.g. when viewing a document and a spreadsheet), they appear to be available concurrently, the user simply toggles between them by using the mouse - there is no need to quit one environment in order to work in another.

The Lisa is object-oriented in this respect.

TopView

The programmes in use are apparent: in using DOS the files necessary for the system commands to work appear in the C: directory listing. In TopView itself the user is required to indicate when a different programme is needed. If s/he wants to edit a document the Personal Editor must be called (or another word processing package). Also the address of the file to be edited must be entered into the programme by the user - selections made in the DOS Services environment do not port automatically into the other programmes.

TopView cannot be considered object-oriented on this dimension.

5.3.7 Summary of Comparison

The consideration of the two devices on the six dimensions shows that the Lisa is strongly or weakly object-oriented in all respects (dimensions 1-6), whereas TopView is only slightly object-oriented (dimensions 2 and possibly 3), its object-orientedness on dimension 3 (syntax) being dependent on user behaviour.
6. Design

6.1 General

A between subject design was adopted such that subjects were assigned to one of two groups: using the Lisa or using TopView. The between subjects design was chosen to eliminate practice effects between the two styles; balancing for order of presentation was not feasible with such small subject groups. In any case, the aim of the study is general observation, rather than experimental rigour. The tasks to be performed were the same for both groups (in fact the same booklet of task descriptions was used throughout). An A A B B B B A A arrangement of conditions was fixed prior to experimentation. [A = Lisa; B = TopView].

The method of data collection was similar to that undertaken by Buckley & Long (1985). A video recording of the task session was made. A mute version of the video was then played back to the subjects and they were asked to give a retrospective protocol. The measures taken were time to complete tasks; number of tasks successfully completed; errors (where errors are taken to be deviations from the most efficient path through the task) and scorings of the preferred command sequence for the TopView group. The contents of the protocols were expected to be informative as to the intentions of subjects and as an aid to inferring the causes of errors.

6.2 Session Schedule

Each session followed the same schedule:

i) Questionnaire - subjects were asked to complete a short questionnaire to ascertain information on individual differences (e.g. sex; age; first language etc.) and their previous experience. The questionnaire appears in Appendix A. The subjects also completed a standard IBM consent form.

ii) Instructions I - subjects were given a set of brief general instructions explaining the nature of the session and to re-assure them about the use to be made of the findings. All instructions appear in Appendix B.

iii) Training - subjects were given elementary training to familiarize them with the device they were assigned to use. The nature of the training is described more fully in a later section.

iv) Instructions II - these instructions explained what the subjects were expected to do in the first part of the test session and gave a description of the task scenario.
v) Test Session I - the subjects were left to perform the tasks described in the booklet. A video recording was made of their performance. This session was limited to 45 minutes.

vi) Instructions III - these instructions explained the second part of the test session: giving a retrospective verbal protocol.

vii) Test Session II - subjects watched a mute recording of their performance and were asked to give a commentary.

viii) Debriefing/Questions - the nature of the study was revealed to the subjects. They were asked a few additional questions relating to their performance and their attitudes/overall impression of the device they were using.

6.3 Details of Video Recordings

All video recordings were made using the facilities of the video studio at IBM Laboratories UK, Hursley.

The computer equipment was set up in a room joined to the video control room by a one-way mirror. Thus the subjects could be observed while they were being recorded. Three cameras were used:

1. To the left of the subject: giving a view of the task description booklet.
2. Centrally above the subject: giving a view of the screen.
3. To the right of the subject: giving an overall picture of the subject, mouse and keyboard

A low band Umatic tape was recorded from each camera. These were combined (at the time of filming) onto a single high band Umatic tape - the final output. Generally the picture from camera 2 was used; but pictures from camera 1 were included whenever the subject began a new task because its task number could be seen and pictures from camera 3 were included when there was a period of inactivity on the screen - such as when the subject was making notes.

A microphone was placed (unobtrusively) above the subject in order to capture any utterances. It also enabled the subject to communicate with the experimenter in cases of extreme difficulty. The experimenter could answer questions over the loud speaker system. The sound recording from the task phase was recorded onto the first channel of the audio track. For the retrospective protocols the subjects saw the tape played back over a monitor. A directional
microphone was used to record the subjects' protocols and the recording was put on the second channel of the audio track. A time code (hrs-min-secs-frames) was recorded onto the video tape.

7. Tasks

7.1 General

The tasks were drawn from the domain of common office tasks. This domain comprises a broad spectrum of activities such as sending messages; creating documents; accessing and organising information; making notes etc. as outlined in Section 3 of this chapter.

The task set consisted of six tasks which centered around the theme of correspondence. The six tasks were independent from one another in the sense that success on a task was not contingent on success on a previous task or tasks. The tasks were roughly ordered with respect to their perceived difficulty with tasks 1, 2, 3 being more straightforward and requiring less steps than tasks 4, 5, 6. A description of each task was typed on a separate page and the pages were arranged into a booklet. Subjects were asked to attempt the tasks in order. The tasks were described at a high level, i.e. in a manner which did not reflect the actual steps necessary to perform the task using the computer.

The tasks were devised to conform with the following considerations:

i) time - it must be possible for the subjects to complete the tasks in the time provided;
ii) authenticity - the tasks must be representative of those that occur in the real-world situation;
iii) equivalence over the two computers - the chosen tasks should be supportable by both computers and the on-line tasks should be as equivalent as possible.

In addition, the tasks were designed to elicit behaviour samples pertinent to the main line of enquiry; this is described in Section 7.3.

7.2 Task Materials

The task materials were restricted to text items (as opposed to graphical or tabular items). All the materials were based on the same scenario and this was presented to the subjects along with the task instructions (see Appendix B)

The task materials consisted of fifteen separate text items (letters, memos, reports etc.)
arranged into three groups. The containers for items were represented by folders on the Lisa and sub-directories on TopView. The three groups were labelled: Current; Info; Letters.

The contents of each Group is given below, beside each item is a simple descriptor of type. The figure in brackets refers to the task where that item was used: "d" indicates that the item was a distractor, i.e. that it was not used in any of the tasks.

The Current Group had 5 items:

- Memo-87                      [1]
- Memo-JS                      memos        [5]
- Memo-DS                      [d]
- Ref-GH                      letter        [4]
- Plan                        notes        [d]

The Info Group had 3 items:

- Figures                      report        [4]
- Prices                       list          [d]
- Organisation                 note          [d]

The Letters Group had 7 items:

- Ref-Comm                     [6]
- Ref-CG                       [d]
- Ref-GS                       [2]
- Ref-MS1                      letters     [3]
- Ref-MS2                      [3]
- Ref-MS3                      [d]
- Ref-WJ                       [6]

Note that in TopView only there was an additional file in each group labelled, "PE.PRO"; this was a profile specification for the Personal Editor. Its inclusion was unavoidable as such a file is necessary for the Personal Editor to access a sub-directory.

7.3 Task Descriptions and Reasons for their Inclusion

The task descriptions are listed below. Following the list, the reasons for using these particular tasks are given.
TASK 1:
Discard an old memo called "Memo-87" which is filed under Current.

TASK 2:
Find the letter which you wrote to a Mr Shore and check what other information you said you would enclose with the letter.

TASK 3:
Find the two letters which you wrote in December and rename them appropriately:
"Ref-KD"
"Ref-PH"

TASK 4:
You are currently writing a letter to Gina Hampshire. To complete the letter, you need to find some information which appears in a report. The relevant report is filed under "Info". Find the information and complete the letter. When complete, file this letter with your other letters.

TASK 5:
Make a copy of the memo called "Memo-JS" and file it with the letters. Re-title the copy "Memo-PJ" and make the following changes:
 i)  address it, "To: Peter Jones".
 ii) change the date to today's date.
 iii) change "Jerry" to "Peter" at the beginning of the text.

TASK 6:
Compare the length of the two letters "Ref-Comm" and "Ref-WJ". File the longer one with your current work. In the longer letter, you have omitted a date for a meeting, fill this in so that the date reads 1st March 1988.

There were a number of reasons why these tasks were chosen, these are described below. In each case the relevant task or tasks is indicated.

i) to observe the preferred command syntax, object => action vs. action => object, for the TopView subjects [Tasks 1-6].

ii) to expose subjects to the different representations of entities on the two devices:
 - as icons (Lisa).
 - as textual labels presented in a list (TopView).
In the Lisa, the folder and document, object types are distinguished by their having different icon designs. In TopView, sub-directories and files can appear in the same listing for a directory and they are perceptually similar - sub-directories are only identifiable by having the tag, '<DIR>' attached [Tasks 1-6].

iii) to compare the two different methods of deletion, via a command word for TopView ['erase'] and via an object in Lisa ['wastebin']. The task, as supported by Lisa, involves a greater object-action ratio than the TopView version. The Lisa method involves two objects, document & wastebin and one action, move, whereas TopView uses one object, file, and one action, erase [Task 1].

iv) to observe the preferred task structure (object-oriented vs. operation-oriented).

Task structure refers to the sequence of operations required to perform a task as in Jorgensen et al (1982) where two task structures were identified:

- object-oriented - each object is dealt with in turn
- operation-oriented - an operation is performed until the number of objects has been exhausted, then the next operation is begun, [see Chapter 3 for more details].

As neither system placed a constraint on which task structure should be applied, this task provided a means for observing the preferred task structure for the user and any relationship between it and the type of system used [Task 3].

v) to compare the errors and difficulties across the two devices in using the word processing environment [Tasks 4,5,6].

In TopView, the Personal Editor programme has to be explicitly invoked by the user for editing, although reading the document is possible from the DOS services programme. In Lisa, accessing the document (for reading) automatically invokes the word processing environment. Also, in TopView there are a number of inconsistencies between the two "environments", DOS Services and Personal Editor, e.g. use of the mouse - any problems stemming from these will be noted.

vi) to observe whether any Lisa subjects treated the two objects (letter and report) as independent and compared them directly on the screen [Tasks 4, 6].

If they are placed side by side, the one to be edited can be activated simply by clicking the mouse button when the pointer is in its window, the new data can then be copied in while the report remains displayed. (this contrasts with enforced sequential performance of the task, where only one item can occupy the screen space at any one time).

vii) to look at the implications of the different methods of filing [Tasks 4, 5,6]
- by Direct Manipulation in Lisa - move the to-be-filed item into the correct folder.
- by Command in TopView - the to-be-filed item must be copied to its new destination and then the original erased. (It is hypothesized that this latter method would require more intermediary translations between what the subjects wants to achieve and how s/he is forced by the device to achieve it via the device - therefore there is a wider scope for errors).

viii) to compare the two methods of copying - TopView requires subjects to remember path names, e.g. the one for the new location of the copy; Lisa only requires the copy to be moved by Direct Manipulation to its new destination [Task 5].

8. Subjects

All subjects were employees of IBM UK Laboratories Ltd., Hursley and their participation in this study was voluntary.

The requirements for subjects were that they should:

- have some general computer experience e.g. use the IBM internal network.
- not be dedicated or expert users of any particular package e.g. Wordstar or machine e.g. IBM PC.
- not have extensive knowledge about computers; programming etc.
- be accustomed to doing office tasks and organising their own time.

People with some experience of computing were chosen in preference to the totally naive in order that errors and difficulties could be taken to be associated with the particular device rather than being more generalised problems concerning computer usage per se. Also, it was desirable that subjects should be fairly confident with computers, so that they would adopt an investigative approach to solving the task problems.

The order in which the devices were to be used was fixed prior to the running of the study. The ordering was chosen to minimise the practical problems involved in moving the equipment and to balance the effect of practice on the part of the experimenter. Subjects were then assigned to the 'slots' on the basis of their availability.

9. Training

Prior, to the test session each subject was given a training session lasting 30-45 minutes (the training was longer for TopView than for Lisa due to there being more to explain). The
The purpose of this session was to familiarize the subjects with the interaction techniques necessary to use the device to which they had been assigned. The training was such that every activity necessary to perform the subsequent tasks was included - thus the subjects would be equipped to perform the tasks, providing that they could remember and combine the activities taught in the training session.

Sample activities using TopView include:

- learning to produce the TopView menu and select Scroll.
- learning to use the function keys for the Personal Editor.

Sample activities using Lisa include:

- learning to move and size a window.
- learning to drag an icon.

The training was in the style of guided "hands-on" experience. Instructions were given verbally by the experimenter so the subjects were told how to interact and practised each element under supervision. The training materials were contained in a folder/sub-directory labelled 'Training' and were only required during the training session.

After all the elements had been covered, the subjects were given a sample task to attempt under pseudo-experimental conditions to ensure that the subjects were sufficiently proficient to begin the task session. The sample task involved locating and re-labelling a document and editing a word within the document.

10. Expectations of the Study

A number of expectations were generated in order to provide a focus for the analysis. These expectations are based on the premise that object-oriented interfaces are easy to use. Combining this premise with the dimensions for determining whether an interface is object-oriented, leads to predictions about how particular interface features might influence performance. The expectations are listed below in a manner which relates them to the dimensions:

**Dimension 1: Representation of task entities**

a) Presentation of task information in the form of objects provides a better mapping with the user's view of the task. Generally, it would be expected that users would make fewer errors
and be able to accomplish the tasks more successfully.
b) The presentation of system entities as objects provides the user with a metaphor for utilizing information about those objects to best effect (e.g. location). It is expected that the Lisa users will have a clearer idea of the arrangement of the system and will make fewer errors of ‘navigation’.
c) An object-oriented interface will encourage users to conceptualize objects as being independent entities. The prediction follows that in Tasks 4 and 6; the documents will be compared directly on the screen for Lisa subjects, while sequential behaviour will be exhibited by TopView subjects.

Dimension 2: Limitation of command set
No expectation was generated from this dimension because both interfaces had a similar rating with respect to it.

Dimension 3: Syntax
a) Communications level syntax:
Given the nature of the tasks, an object => action syntax would be preferred. This expectation predicts that TopView subjects will use object => action syntax; although this syntax has never been demonstrated.
b) Task level syntax:
An object-oriented task structure is more ‘natural’. The task structure will be observed for Task 3, it is expected that an object-oriented structure will be preferred, although TopView encourages an operation-oriented structure by the nature of its screen layout and cursor movements.

Dimension 4: Object : action ratio
As object-oriented interfaces are suspected of having more objects than actions, a method of task support which involves the use of a greater ratio of objects to actions is expected to facilitate user performance. In Task 1, where the Lisa supports the task using a greater object:action ratio, subjects would be expected to perform faster and with fewer errors.

Dimension 5: Representation of properties
As it is unclear how the systems differ on this dimension and the tasks are not expected to bring any differences into play. No expectation was generated.

Dimension 6: Concealing the mechanics of the programme
An object-oriented interface protects the user from the raw details of the programme; so letting him/her concentrate on doing the task. As the word processing facilities are automatically invoked in Lisa and there is no necessity for users to specify details of file
addresses, it is expected that the users will make less errors (Tasks 4,5,6).

11. Analysis

The results are presented in three basic sections. The first section deals with findings from the questionnaire and gross measures of task performance: time; number of tasks completed; interventions. The second section presents the errors made by the subjects and is followed by a discussion. The third section focusses on the results concerning syntax and is followed by a discussion. A statistical treatment of the data was not warranted due to the exploratory nature of the study.

11.1 Section 1: General Measures

11.1.1 Questionnaire

The purpose of the questionnaire was to elicit some personal details from the subjects and to find out about their use of computers at work and at home.

*Personal Details*

The questionnaire revealed that all subjects were of British nationality and spoke English as their first language. Only two subjects had any knowledge of other languages. All subjects were qualified to at least A Level standard. The ages of the subjects ranged from 19 to 57 years with an overall mean of 31.75 years.

The mean ages of subjects by Group were:

<table>
<thead>
<tr>
<th>Group</th>
<th>Mean Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lisa</td>
<td>40.5 years</td>
</tr>
<tr>
<td>TopView</td>
<td>23 years</td>
</tr>
</tbody>
</table>

There were three female subjects and five male subjects, distributed as follows:

<table>
<thead>
<tr>
<th>Group</th>
<th>Sex Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lisa</td>
<td>2 females; 2 males</td>
</tr>
<tr>
<td>TopView</td>
<td>1 female; 3 males</td>
</tr>
</tbody>
</table>

It was unfortunate that the age and sex distributions for the two groups were not more similar. However, this effect occurred because (as mentioned earlier) the machine order was invariant and subjects were assigned to Groups on the basis of their availability irrespective of their age or sex. However, there is no reason to believe that age and sex have any confounding effects for the purposes of this study.
As for their job details, each subject came from a different department so a wide range of job titles was covered:

- Facilities Engineer
- Human Factors Engineer
- Librarian
- Personnel Trainee
- Project Engineer
- Secretary
- Technical Writer
- Trainee Accountant

The time spent in these positions ranged from 3 weeks to 7 years. All subjects used computers for their work and the length of their experience varied from 4 months to 25 years. The subjects all reported using their computers daily or on most days. In the main, the subjects' computer use was restricted to terminals on the IBM internal network. Three subjects, (S3, S6, S7) had used the IBM PC.

Use of the computer at work was mostly confined to everyday office tasks: sending and receiving notes; filing; keeping a diary; generating text using a word processor. Some subjects used computers to support specific activities involved in their jobs such as data logging, on-line ordering and data retrieval.

Only one subject (S7) used a computer at home. He has used an Amstrad 6128 over the last six months for playing games and learning BASIC programming.

In summary, it seems that the subject group conformed well to the ideal subject specification. All subjects used computers to support their daily work, while the content of their work was not computer related.

11.1.2 Time

It would not be sensible to attach great importance to time measures between groups in this study as the differences between machine response times and the different methods they provide for supporting tasks would account for substantial portions of the variance. However, gross time measures will be reported so that general trends in performance speed can be noted.
Overall Time to Complete Tasks
(measured only if < 45 minutes: subjects were stopped after 45 minutes)

<table>
<thead>
<tr>
<th>Subject No.</th>
<th>Lisa Group</th>
<th>TopView Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>45</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>24:51</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>30:55</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>44:22</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>32:12</td>
<td>45</td>
</tr>
<tr>
<td>4</td>
<td>45</td>
<td>45</td>
</tr>
<tr>
<td>5</td>
<td>32:09</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1
Overall time taken to complete tasks

Three subjects (S1 [Lisa]; S4, S5 [TopView]) were stopped after 45 minutes. The point reached in the tasks by these three subjects will be described in a later section when performance profiles will be given for each task.

These overall times show that the two groups were well matched in terms of their performance times. In each group, there were two quick subjects and two slow subjects.

Times taken for each task
The times taken by subjects to complete each task (measured to the nearest second) are shown in Table 4.2.
### Table 4.2
Time taken to complete each task

By inspection there seem to be no major differences between groups. There are clear trends showing Task 4 to take longer to complete, than Tasks 1, 2, 3. There seems to be some practice effect, as despite being fairly complex Tasks 5 & 6 are accomplished much faster than Task 4.

Task 1 seemed to take a surprisingly long time for Lisa subjects to complete. This fact will be discussed later.

#### 11.1.3 Number of Tasks Completed

The number of tasks that were completed by the subjects in both groups appear in Table 4.3. For a task to be considered complete the task materials should be in the end-state described by the task description when the subject finishes the task. E.g. if the task description said, "file the letter with your current work", then the letter should appear, appropriately named, in the Current folder. (Note that task 4 has been divided into two parts: the editing part and the filing part).
Table 4.3
Number of tasks attempted by subjects

TOTALS (by Group)
Lisa: 23 tasks completed out of a possible 28.
4 tasks attempted but not finished in a satisfactory state.
1 task not attempted.

TopView: 18 tasks completed out of a possible 28.
5 tasks attempted but not finished in a satisfactory state.
5 tasks not attempted.

Although no strong conclusions can be drawn from such data, the general trend would seem to be that though there were little differences between groups with respect to time; Lisa subjects completed more tasks (where task completion is determined by arriving at a satisfactory end state). The major difference between groups appeared in the later tasks where filing/organization of information was required. In particular, the second part of task 4 was not attempted by three TopView subjects and was not completed successfully by the other TopView subject.
11.1.4 Interventions

Subjects were instructed that if they experienced any severe difficulties they could ask out loud and the experimenter would answer over the loudspeaker system - this was to avoid the situation where a subject has repeated problems on one task which prevent him/her from trying the other tasks and no new behaviour is observed. The number of interventions provides another gross measure of task performance. Obviously this measure lacks rigour as there are personality differences between subjects in their propensity to ask for guidance and in a study such as this they cannot be balanced. Also, there is no measure (other than a subjective one) of the severity of the problem leading to the need for intervention.

A list giving details of all interventions appears in Appendix C.

Comparing the number of interventions for the two Groups gives:

Lisa: 6 interventions
TopView: 15 interventions

Lisa
Of these, two instances were asking for information about how to carry out the task, rather than problems with the device (S2; S8 [TASK 2]).

TopView
Two instances concerned information about task strategies (S4 [TASK 4: no.2]; S6). However, the intervention for S6 also included a reminder about the presence of the Current sub-directory. The remaining requests for help concerned specific problems that subjects experienced with the device.

It seems that TopView subjects generally needed more information about the precise file-names and path-names. They were unable to work these out for themselves.

11.2 Section 2: Error Data

11.2.1 Error Data by Task

A fairly wide definition of errors was used. Errors were scored by comparing the video recording of each subject's performance with a template for the shortest path through the task. Any deviations from the shortest path were counted as errors. Information from the
retrospective protocols was used to infer possible causes or interpretations of errors. As TopView allowed subjects to use an object => action syntax with no negative consequences, this was not scored as an error, but was counted as an additional metric i.e. number of instances of object => action syntax. A relatively high number of instances of "incorrect" syntax usage was found for TopView subjects and these results are reported in the next section.

A description of each group's performance on each task is given below. The sources of error are described.

Listings of the actual errors scored are given in Appendix E, while the templates for the shortest paths through the tasks (given the level of the training) are contained in Appendix D.

**TASK 1**

**LISA**

This task was completed by S1, S2, S7. Subject 8 failed to complete the task successfully as the Memo was left (Set Aside) on the desktop.

For three of the subjects in this group (S1, S7, S8), this task did not serve its purpose of inspiring confidence. They took longer than expected to complete the tasks and their performances were error-prone. These three subjects seemed to forget how to discard by direct manipulation, instead they attempted to find some discard option in a menu. Additionally, S8 had problems with the mechanics of direct manipulation i.e. dragging icons and also made confusions over the functions of objects e.g. attempting to put the wastebin into a document and treating a folder containing a document as the document. Subject 1 and S8 tried to select shadowed icons which suggests that they failed to grasp their function.

Subject 7 omitted to select the intended object and so received unexpected menu options (the wastebin was highlighted when he wished to work in the Current window).

**TOPVIEW**

This task was completed by S3, S4, S5, S6.

In contrast to the Lisa subjects, subjects in this group finished the task quickly with the only errors being due to mouse use and unnecessary checking behaviour, i.e. opening the memo to see if it was the right memo.
TASK 2

LISA
Task completed by S1, S2, S7, S8.

This task did not cause any major problems for subjects in Group 1. Subject 1 experienced some problems due to an inadvertent ‘Set Aside Everything’ command. She attempted several recovery strategies because she found the disk icon to be unsuggestive of its contents.

TOPVIEW
Task completed by S3, S4, S5, S6.

Initially, three subjects had problems over the procedure for changing directory (going a step back up the hierarchy): it requires changing the path name at the top of the directory panel. Subject 3 had a very minor problem - more to do with the mouse than the procedure. Subjects 4 & 5 had major problems and became confused with the idea of switching, which is usually for changing programmes or windows, within a programme. Subject 4 required an intervention while S5 adopted a strategy which became problematic later on - opening another copy of DOS Services. Subject 4 also had a problem of not knowing how to display an item on the screen the ‘Type’, command seemed unsuggestive of its function. It would seem here that mechanics of navigation for TopView presented problems though they are probably due to the required procedure than the fact of textual representation.

TASK 3

LISA
Task completed by S1, S2, S7, S8.

This task did not cause many problems. Subject 2 and S8 both had an error-free performance. Mistakes often arose from lack of mouse control and recovery from these was easy. Subject 1 had a few problems due to her strategy of looking for some kind of log file which would record the dates when work was produced. Also the editing functions caused problems for S1 which seem to be attributable to an effect of her previous experience where the Delete function operated in an opposed direction.

As for the use of task structure, three subjects (S1, S2, S8) used an object-oriented structure while S7 used an operation-oriented structure.
TOPVIEW
This task was completed by S3, S4, S5, S6.

There was no need for subjects to change directory in this task and it was accomplished easily.

A novel strategy was employed by two subjects (S3, S5 for half the task) who made use of the Date data although this had never been pointed out. The strategy worked in this case, but is not foolproof as the dates are updated for every modification whereas the date, appearing as part of the letter text, may not be changed.

As for task structure, the results of two subjects (S3, S5) cannot be counted due to their different strategy; S4 used an operation-oriented structure while S6 used a mixed strategy - beginning in an operation-oriented manner, but then S6 decided to check the contents of the first letter for the initials and reverted to an object-oriented structure.

Overall, the task structure results seem to be inconclusive, though there is some trend towards using a structure consistent with the type of syntax used (NB. Subject 6 used an object-> action syntax throughout) - i.e. an object-oriented structure with an object => action syntax.

TASK 4: PART I

For the purposes of analysis, this task has been divided into two parts: Part I involves finding the relevant information and fitting it into the letter and Part II re-filing the letter in a different folder/ sub-directory.

LISA
This task was completed by S1, S2 and S7. Subject 8 accessed the information and the letter, but put fictional figures in the letter, thus simulating the task.

Many of the errors for the Lisa group were concerned with task strategy rather than interaction with the computer. Three subjects (S1, S7 & S8) made a task error at the beginning of the task - looking in the Information folder first, before locating the letter. All subjects initially looked in the Letters folder for the letter, but they recovered from this fairly easily by opening the Current folder.

During execution of the task only S2 used the strategy of opening both letter and report on the screen and then just copying the figures from one to the other. Subjects 1 & 7 used a sequential strategy whereby they made notes from the report and then copied them into the letter.
Before deciding to make notes, S1 was preoccupied with looking for some kind of copy function which would allow the required text to be selected and copied across. Of course, the Lisa does allow this (Copy and Paste), but it is a fairly advanced editing function which was not demonstrated in the training. Subject 1 also had problems with the editing environment.

**TOPVIEW**

This task was completed by S3, S4, S5 and S6.

As expected, this task caused more problems for the subjects. Like the Lisa subjects, all the TopView subjects made the task mistake of accessing the Information folder/sub-directory first. Similarly, all subjects initially looked for the letter in the Letters folder/sub-directory. Here, recovery was not so swift as the subjects seemed to have forgotten about the existence of the Current folder/sub-directory.

Changing directory still caused problems for three of the subjects - mostly typing mistakes. It seems that the subjects in this Group frequently swapped between directories - more than was necessary. Often swapping seemed to be a consequence of subjects losing their place. Several subjects looked for the letter in Letters again after they had previously found it in Current. They did not seem to have built up a mental picture of where items are located. The number of (fairly irrelevant) steps the subjects must undergo to traverse through the system seems to have an effect on their ability to do the task efficiently. Subject 6 forgot the name of the letter and had to go back and check in the Current directory; S5 forgot the information after he returned to the letter and had to go back to the Info directory again.

It was interesting that two subjects (S3, S4) did make an attempt to edit the letter from within the DOS Services environment. Their behaviour suggests that they considered the letter to be an object on which legal actions could be performed, without regard for the use of different application programs.

Having to invoke the Personal Editor caused a great many problems - firstly in accessing the programme (S4, S5) and then in entering the details of the file into the panel (from their comments subjects found it irritating that though they had selected the letter while in DOS services this information could not be transported across to the editing environment). Subjects seemed to be uncertain about the format required and the actual names and locations of files. The inconsistencies between DOS Services and the Personal Editor caused problems particularly for S5 who tried to use the TopView menu when most of its functions become unusable.
TASK 4: PART II

LISA
This task was completed by S1, S2 and S7. Subject 8 attempted the task but did not complete it successfully.

Three subjects in this group performed this part of the task with no problems; they appeared to have mastered the art of direct manipulation. Subject 8 appeared to forget about how to file using direct manipulation and set about finding some way to file using menus. She experienced some confusion over the function of 'Setting aside'. Also, when an attempt at direct manipulation was made, the subject tried to move the whole folder rather than the document it contained.

TOPVIEW
This task was attempted only by S3.

The performance of the TopView subjects was surprising as only one subject attempted this part of the task, S3. The other three subjects made no attempt and made no mention of having omitted it in their protocols.

Subject 3 did not complete the task successfully. He sent the copy to the wrong destination and he failed to erase the original. Thus, it was copied rather than moved. It seems that he failed to translate the task correctly, moving vs copying. However, he said later that he decided not to erase the original which suggests that he was uncertain that the operation had been performed successfully. If the subject was confident that the file was in the right place, there would be no problem over erasing the original. The subject did not seem to be aware that the file was in the wrong directory which suggests that there may be a problem of visualising the location of items for subjects using TopView.

TASK 5

LISA
Task completed by S2, S7 and S8 - S1 was stopped during the editing part of this task (she did go on to complete it successfully after time).

This task did not cause too many problems for the Lisa subjects - particularly the copying part. Subject 8 suffered a number of problems in trying to re-file the copy: initially forgetting direct manipulation and trying to find some filing option (setting aside). Also, she displayed confusion between a folder icon and a single document: trying to move the entire folder. There
were no editing problems for this group.

**TOPVIEW**

This task was completed by S6. Subject 3 attempted the task, but edited and renamed the original and sent the copy to the wrong location: placing it in the TopView directory and naming it "Letter".

Both S4 and S5 did not have time to complete this task.

Subjects experienced many problems with this task, the problems occurring at two points: specifying the new location during copying and accessing the file using the Personal Editor. The subjects realised by now that editing had to be done via the editor - but specifying the file address was problematic. They seemed to be confused about the path name of the original file and how this should be changed to indicate a move to the Letters directory; and they chose an assortment of permutations which caused problems, because the address was often legal, but not correct for the task and subjects did not realise this. Subject 3 may have mis-interpreted the task as he renamed and edited the original file. The copy was sent to the TopView file and was renamed Letters: this address had been used (erroneously) before in Task 4.

**TASK 6**

**LISA**

This task was completed by S2, S7 and S8 (S8 had considerable help in the form of interventions).

A minor problem of inadvertently de-selecting an icon was experienced by S7, otherwise all errors were suffered by S8. She had many problems with filing and had two attempts, before and after editing the letter. Two features of the performance were noticeable: a) a tendency to try to act on a whole folder rather than a particular item; b) treating 'Save and Put Away' as some method of filing - although the Put Away means that the object is put in its original position.

Only one subject (S2) used the strategy of comparing both letters on the screen at once. The other subjects mentioned in their protocols that they thought this should be possible, but they deferred to their 'safe' strategy of opening and closing things. This observation demonstrates the attractiveness of the independence of objects.
TOPVIEW
This task was completed by S6. It was attempted by S3 who omitted the last step - erasing the original Ref-Comm from the Letters file.
It was not attempted by S4 & S5 who were out of time.

The subjects did not have any major problems, however, some deviations from the ideal path were scored. At the beginning of this task, one of the subjects, S3, decided to correct a former error which had come to light. Seeing the file, 'Letter' in the TopView directory, he copied it to the Letters directory; erased the original and renamed the copy, "Memo-JS". This file was his copied version from the previous task. Both subjects used the same unusual strategy for this task: utilizing the size listing; incidentally, this need not have been correct for the letter with the largest overall size does not necessarily have the longest body of text (it could contain very large addresses) and conventionally letter length is considered to refer to the body of the letter. Subject 3 did not erase the original letter in this case, but the reason for this is not apparent. Use of this strategy meant that the intended comparison could not be made.

However, this task does show that filing causes more problems for TopView where the end result has to be carefully translated into steps e.g. copy; then erase original etc. whereas in Lisa the end result is achieved directly.

11.2.2 Discussion

In this section, the information gained from the error data will be discussed in the light of the expectations set out in Section 10.

Expectation 1a
The expectation concerning the first dimension predicted that the Lisa subjects would be able to accomplish the tasks more easily and would make fewer errors. Evidence to this effect was found: the Lisa subjects completed more of the tasks than the TopView subjects, and asked for help on fewer occasions. During performance of the tasks, they made fewer errors, particularly during the more complex tasks (4,5,6). A large number of the errors they did encounter were fairly trivial in nature e.g. mouse errors such as inadvertently dragging icons; clicking on shadowed icons etc. and recovery was easy.

Expectation 1b
This expectation also predicted that Lisa subjects would make fewer errors of navigation due to more information about location being available. In comparing the error profiles of the two
subject groups, this expectation was found to be supported.
The TopView subjects experienced problems of knowing where they were (i.e. which
directory) in the system and where items of interest may be. In Task 4, in particular, subjects
repeatedly went back to the wrong directory to look for something because they had forgotten
where they had seen it before. In copying tasks, they often sent files to the wrong destination
- which indicates that the path name method of specifying location is difficult to grasp.
It was noticeable that at the beginning of Task 5, Subject 6 was already in the Current
directory, but not noticing this, she proceeded to return to the (main) TopView directory and
reselect the Current directory. This strategy suggests that she preferred to keep to a rigid
procedure for performing the tasks even when it was unnecessary.

She commented:
S6 - 22:10

"I didn't realise I was there so I went back into TopView"

The navigational problems encountered by TopView subjects could be explained in two ways:
i) there is very little to discriminate perceptually between the different directories as they
all appear as lists. The only variation is the length of the list. The command line at the top
of the panel does present the address of the current directory, but subjects did not seem to
derive any benefit from it.
ii) the system does not encourage the subjects to form an overall view of the components and
their relationships, and so behaviour is highly proceduralized.

*Expectation 1c*
The third part of the expectation stated that in Tasks 4 and 6; the Lisa subjects will compare
the documents directly on the screen, while sequential behaviour will be exhibited by
TopView subjects. There was little observed evidence for this prediction, since only Subject 2
used this facility. However, the other Lisa subjects mentioned in their protocols that they
thought this strategy to be attractive, but that time constraints led to them opting for a safe,
known strategy.

*Expectation 2*
No expectation was generated for the second dimension.

*Expectation 3a*
Communications level syntax: the results concerning this expectation are presented
separately in Section 11.3.

*Expectation 3b*

Task level syntax: this expectation predicted that an object-oriented structure would be preferred in Task 3, despite the fact that TopView encourages an operation-oriented structure by the nature of its screen layout and cursor movements.

In Task 3, it was found that three Lisa subjects used an object-oriented structure; only two of the TopView subjects used an appropriate strategy - one of these used an operation-oriented structure and the other used a mixed structure which started as operation-oriented and ended as object-oriented (note this subject used an object => action syntax throughout). Although no firm conclusions can be drawn from these data, they suggest that a task structure consistent with the Communications Level syntax is preferred. Findings were supportive of this suggestion for five out of six subjects.

*Expectation 4*

The expectation for the fourth dimension, object:action ratio, stated that in Task 1, where the Lisa supports the task using a greater object:action ratio, subjects would be expected to perform faster and with fewer errors. Evidence was not found to support this expectation. The method of deletion involving two objects (document and wastebin) was not straightforward for the Lisa subjects (in spite of their training). Only one subject (S2) performed this task without error. It was clear from the protocols of the other subjects that they were looking for a menu option for the action, 'delete'.

E.g.

*S7 - 1:06*

"I was looking at that menu and there was nothing that said anything about discard, so I thought, well maybe 'Set Aside', is a possible discard . . "

This finding throws some doubt on whether using two sorts of objects, i.e. application objects (e.g. document; drawing; spreadsheet etc.) and function objects (e.g. wastebin; printer etc.) is an appropriate way of implementing the task.

However, despite their initial difficulties all Lisa subjects expressed favourable opinions concerning Direct Manipulation in their protocols.

E.g.

*S1 - 1:36*

"... it's actually simpler really when you stop yourself from looking for more
complicated things, a lot of things are just actually moving it and I think I realised that I had to get it so it was all visible and then just move it."

S7 - 7:34

"It was funny using the mouse I'd like to do it more . . . it's much more direct than using the pf keys, you can quickly get used to picking them up and moving them around."

Expectation 5
No expectation was generated for this dimension.

Expectation 6
The expectation concerning the extent to which the 'programme' is concealed from the user predicted that, as the word processing facilities are automatically invoked in Lisa and there is no necessity for users to specify details of file addresses users will make less errors (for Tasks 4,5,6).

The analysis showed that the TopView subjects did make more errors than the Lisa subjects during tasks requiring editing.

TopView subjects found having to switch programmes frustrating, once having identified the object, they wanted to carry on and use it.

E.g.
S3 - 14:40

"I was going to try and edit from within this set - I was going to go into Current . . . 'cos I was trying to go into Letters again and remembered that it wasn't in Letters and I was going to try to edit from here. At this point, I realised that I couldn't do that and I'd have to get out of there and move on . . ."

In addition, the inconsistencies in interaction style between the two TopView programmes made it more obvious to the subjects that they were working with different programmes.

The automatic invocation of programmes in the Lisa facilitated the performance of the subjects using it. However, it is not clear whether automatic invocation of programmes helps subjects because it aids them in sustaining some sort of object concept, or whether it simply makes the task less complex (i.e. it has less steps).

The error data enabled a picture to be built up of the two groups of subjects doing the tasks. By comparing the performances of the two groups it could be established whether the general expectations of Section 10 were offered any support. All but one (concerning Dimension 4) received some degree of support, so it would seem that the dimensions posited do have some consequences for user performance and the nature of those consequences has been observed.
11.3 Section 3: Syntax at the Communications Level

11.3.1 Results Concerning Syntax

In the dimensions characterizing the device, syntax was suggested to be a potentially useful factor for determining whether an interface is object-oriented. The criterion was stated that object-oriented interfaces have an object => action syntax (at the Communications Level). If object-oriented interfaces are easier to use than their contrast, then it is reasonable to expect that their feature, object => action syntax, might have some part to play in this ease-of-use. As TopView allows both object => action and action => object syntax, it was interesting to record whether subjects opted for an object => action syntax (potentially easier to use), despite never having been told that this sequence is legal.

There was a surprisingly high incidence of object => action syntax usage, despite the following dissuasive factors:

i) Screen Layout: the command (actions) panel appears to the left of the directory (objects) area - given the fact that speakers of English read from left to right, this arrangement would tend to promote an action first, object second sequence.

ii) Pointer positioning: in confluence with (i) after any function has been performed (e.g. changing directory; copying a file; returning from the Personal Editor etc.) the pointer returns to the left-hand command box - to avoid choosing an action the user is required explicitly to move the pointer, by means of the mouse, across to the directory box. To use an object => action syntax, the user must expend additional effort.

iii) Training: the possibility of using an object => action syntax was never demonstrated to any of the subjects during the training.

Results (TopView subjects only)
The number of instances of an object => action syntax was determined for each subject in the TopView Group and this is expressed as a percentage of the total number of command sequences performed in Table 4.4, below:
Mean % of object => action instances for TopView subjects = 60.75%

Table 4.4
Percentage of instances of object => action syntax (TopView subjects)

Occasions where S5 used the 'Type' command for inspecting the contents of directories (as in 'Type-Current') are included. Using the 'Type' action for directories is entirely unnecessary. If these instances are discounted, then the percentage of object => action rises to 21.05% for S5.

The number of action => object instances might be slightly inflated for the following reason. If an action is selected at an inappropriate moment, and is disregarded and other activities are performed, it remains selected and is then applied to whichever object is selected next. The sequence is illustrated below in Figure 4.3.
This sequence results in the subject performing action X on object Y because the action X has remained selected.

In this case the subject may have expected that the initial action would be cancelled (at Step 2) and so the subject may have intended this sequence:

This sequence results in the subject performing action Z on object Y.

An example of this are found in the performance of S4 (between 36:14 and 36:20)

E.g.

- **PRINT**  
  Subject selects action by mistake

- **(Intermediary event)** CHANGE DIRECTORY SELECT CURRENT
  Subject assumes that the action has been cancelled and goes on to change directory

- **MEMO-JS**
  Subject selects the object he wants, intending to Copy it, using o => a syntax: Memo-JS / Copy...

- **PRINT MESSAGE BOX APPEARS**
  ... but the Print command is still selected and is executed, referring to the next object (Memo-JS) giving the appearance of an a => o syntax: Print / Memo-JS.

**Figure 4.3**

Example of an unintended action => object syntax sequence
11.3.2 Discussion

At least two reasons can be suggested for the TopView subjects' preference for the object => action syntax.

i) object => action syntax fits better with the subjects' method of performing the task and their way of thinking; it is more "natural", "intuitive" etc. Arguments along these lines have been advanced in support of object-oriented interfaces, in general.

ii) the use of object => action syntax arises from an inconsistency of the TopView interface. Sub-directories are selected directly from the Directory Panel, using Mouse Button 1 and this results in their contents being displayed on the screen. Subjects may persist in this method of interaction when they want to see the contents of a file (this should be achieved via, 'Type / filename'). Both filenames and sub-directories can appear in any directory listing and they are represented textually, the only difference between them being the presence of the label, <DIR>, for directories. Under this expectation, subjects do not make a firm distinction between files and directories and so may expect the same methods of interaction to be applicable to both.

Quotes from subjects (from retrospective protocols) are seen to offer support for both arguments:

In support of i):

S3: On being asked about using object => action syntax,

2:48
   "... Yes, I tend to do things often as coming from ... like you're working your way through a book backwards - you tend to go to the index and then to the bit you actually want to look at in some ways ..."

3:25
   "I think it's possibly something that I've done before when I've worked. I've used other systems, you know, like WIMP systems before and I found that what would happen would be you would choose what you want and then go and choose what you are going to do with it, rather than choose what you are going to do with something and then choose what you are going to do it with, the other way round."

4:06
   "Looking at it, it looks pretty cock-eyed, I realise that it doesn't look right, but it felt right to choose and then choose what you were going to do, rather than choose what you're going to do and then choose what you're going to do it to."

S3: Noting the influence of dissuasive factors -
"You can see here I'm doing it backwards again although I think I did it forwards sometimes, basically because the cursor was sat over there."

S6: During Task 1 - erasing a file -

00:57

"I couldn't remember which order to do it in, so I thought I'd pick that [file] first and then go back and erase it. Instead of erasing it first."

In support of ii:

S5: When asked whether he realised that he was using o => a syntax:

26:32

"Yes, I did, I thought, well, on the first one [directory] I'd realised you can just type Button 1, I tried it on this [file] and it flashed up 'Type' so I realised that then you had to press in 'Type' as well."

When asked further about his expectations of looking in a file by pressing Button 1

"... I didn't see why it shouldn't because it's on the same panel." (as the directory)

Obviously it is difficult to establish whether it is i) or ii) that accounts for this use of object => action syntax, or indeed both.

However, if it were wholly a case of ii) then one would expect object => action syntax to occur only when 'Type' commands (those that result in opening/ displaying on the screen) are used. Here it was found that S6 used object => action in 100% of instances, using it first for an Erase function and Subjects 3 & 4 both used it for copying. However, due to the nature of the tasks a large number of the object/action pairings involved 'Type', so it is difficult to make a fair comparison between the number of object => action occurrences for 'Type' as opposed to other commands (such as 'Rename', 'Copy', 'Erase' etc.)

The fact that an object => action syntax was used 60.75% of the time despite there being a number of factors which do not encourage its use, suggests that an object => action syntax is desirable under some conditions. So there is some evidence for the prediction that object => action will be preferred. However, the nature of the conditions under which it is preferred is, as yet, uncertain i.e. whether object => action is only useful in instances where it serves to raise the consistency of interaction (on the lines of argument ii) or whether it is more generally useful. Clearly, there is a need for investigation.

There is a need to establish whether object => action syntax is generally preferable or
whether (and how) its usefulness is conditioned by other factors such as:

i) task context (the nature of elements to be implemented may determine a suitable syntax);
ii) task structure (i.e. whether the ordering of the elements of the task need to be reflected in
the syntax at the Communications Level);
iii) semantics (object => action a may be particularly preferred for certain operations [where
the differentiation between operations is semantic] such as Open/Look at/Type, but other
operations may be best implemented by an action => object syntax);
iv) components of systems i.e. the ratio of objects to actions.

In addition, it would be interesting to establish why an object => action syntax is preferred. A
plausible suggestion is that in a particular task context, the objects may be presumed to be
novel (that is the particular instances of objects NOT the generic object types such as
Document; Memo; Letter etc.) and to have a strong relationship with the specifics of the task
and so are easily identified as being related to a particular task. For example, in writing a
letter of complaint the crucial factors are:

- that the letter is addressed to the relevant party
- that a suitable expression of disquiet is included in the text
- that it has sufficient details about the object of complaint

A particular letter may contain these and will be distinguishable from the next letter, say
one of thanks to an ageing relative, by its content - and though the structure, medium and form
of both letters are equivalent and are recognised to be such by the user (classification of items
at the Basic Level) they will be very different at a more detailed level and so are
distinguishable. The actions, on the other hand, e.g. to copy, print out or delete a letter have
exactly the same operation in every instance and the set of applicable actions is common to
to all instances of a particular object type. Perhaps actions are conceptualized at this higher
level (akin to the Basic Level) as distinctions between them seem to be made on a temporal
basis. The argument runs then, that this relationship between the content of instances and the
task the user is trying to achieve has an effect such that the user prefers to pick the object
first - in order to ensure a correct selection and then perform some standard action.

12. Summary and Conclusions

Of the four dimensions which had expectations and predictions associated with them, all
were found to have some consequences for user performance; though not always in the expected
direction.
These findings were generated by the study:

- that Lisa is easier to use than TopView;
- TopView users had problems with:
  - navigation and forming a view of the system;
  - having to change programmes to edit documents;
  - the inconsistencies between the programmes;
  - having to specify location information in terms of path names.
- Lisa subjects had problems with:
  - direct manipulation (initially)
  - working out the referents of icons (particularly the disk icon)
  - performing a task which involved a greater object: action ratio
- TopView subjects prefer to use object => action syntax in 60.75% of instances

If the objectives stated in Section 2 are considered it appears that this comparative observational study has been successful in achieving its purpose. The dimensions which were used to distinguish between the devices selected did have some observable consequences for user performance. In all but one instance, it was found that the object-oriented value on the dimension was associated with ease-of-use.

The examination of the preferred syntax metric (for TopView subjects) showed that object => action syntax was preferred in the majority of instances. This suggests that syntax (at the Communications Level) is a factor worthy of further experimentation, as the conditions governing the preference could not be derived from this study. The study has also been successful in providing information about the device features which have potential for being manipulated in future experimental work.

The study suggests at least three further directions for investigation.

The first is to study syntax (at the Communications Level) in more detail, with a particular emphasis on investigating the conditions which influence its use. The second direction is to explore the feature of automatic invocation of programmes. This was seen to have a large effect on performance, but the underlying reasons for this could not be determined here.

The third direction is an investigation into the expression of location information. It seems that using objects does enable the user to grasp more information about location but a number of factors could contribute to this effect e.g. in iconic interfaces the user plays a more active role in determining the location of items; the object display may allow the user to build up a better model of the device world etc.
The direction selected for the investigation reported in the thesis will be described in the next chapter.
Chapter 5
Method for Testing

1. Introduction

The purpose of this chapter is to set the scene for the major portion of the empirical work which is to be reported in Chapters 6, 7 and 9. It draws upon the findings of the investigation so far, in order to argue for the adoption of a particular empirical approach.

First, a rationale is presented for conducting experiments which focus on a single dimension.

Next, the identity of that single dimension is decided. The decision being guided by the results of the previous analytic and empirical activities. The dimension, syntax at the Communications Level, is further described and a rudimentary User Model is proposed.

Finally there is a description of the method which will be employed for exploring the dimension, syntax, under controlled conditions.

2. Rationale for Single Dimension Experiments

The previous chapters (Chapters 2, 3 & 4) have offered the premise that object-orientedness is multi-dimensional i.e. it does not reside in a single factor, but, instead it can be characterised by a combination of features on a number of relevant dimensions. It has also been suggested that object-orientedness is a continuous property, in that devices can be judged to be more or less object-oriented according to their particular configuration of features on the relevant dimensions.

In Chapter 4, it was found that comparing two devices which contrasted with respect to their object-orientedness, was fruitful for gaining a general picture of the user consequences of the different device styles. However, the study had its limitations: it was unable to provide detailed information about the effects of particular device features. While comparative observational methods are useful for demonstrating the likelihood that a particular feature has an effect (positive or negative) on user performance, they can neither confirm that effect nor can they provide an explanation for the existence of the effect. In short, they are unsuited to the provision of statistical evidence and enabling causal relationships to be inferred. Clearly, one can speculate on possible causal relationships given the results of the study, but any explanation in terms of cognitive psychology can only be guided intuitively as the observational study does not lend itself to rigorous hypothesis testing.
The limitations of observational studies (such as that reported in Chapter 4), are due, in part, to the size of the set of variables which are present. In this particular observational study, the two devices differed in a large number of ways, some of which were of interest i.e. their effects were being compared and analysed (such as the degree of automatic invocation of application programmes and the command syntax) whereas others were incidental e.g. the TopView screen presents text in green while Lisa has black text. These incidental variations offer a potential for confounding effects. Since the degree of variation between the two devices was high and the number of subjects low, there was little opportunity for exerting control over all these extraneous factors.

It has been noted that the problem of confounding variables is not unique to the two devices selected for the observational study. Indeed, even within the set of extant devices which would be classed as object-oriented there is much variation. The empirical activity of this investigation hopes to provide some reliable findings concerning device features that are pertinent to the classification of devices and interfaces as object-oriented. To this end, it is necessary to perform controlled experiments which test the effects of the device features of interest.

The strategy adopted in this investigation will be to assess particular dimensions in isolation, i.e. to compare two features on a dimension of interest. Although it is recognised that this strategy will lead, inevitably, to the lowering of external validity, it is desirable, because it reduces possible confounding effects and enables a greater degree of control to be exerted.

3. Selection of the Dimension, Syntax

Given that the previous section has argued for examining particular device dimensions in isolation, the question remains as to which dimension should be selected as a focus.

First, the requirements for the dimension will be described. The first requirement is that it should be one of the dimensions that is thought pertinent to the classification of devices as object-oriented. Secondly, it should have consequences for user performance i.e. there should be reason to believe that varying a device's features along the dimension should affect the ease-of-use of the device.

In order to satisfy the second requirement: that the dimension should have consequences for user performance, the findings of the observational study will be reviewed. The dimensions which were found to have consequences were:
- syntax (possibly both at the Communications and Task Levels)
- automatic invocation of application programmes
- presentation in terms of objects, as a means of aiding navigation

(Note that the observational study did not permit all the proposed dimensions to be compared and it is reasonable to expect that the other dimensions e.g. representation of object properties, will influence user performance in some way. However, empirical investigation of all the dimensions is beyond the scope of this investigation).

From these, the dimension syntax (at the Communications Level) will be selected as the focus for future experimentation. It fulfils both the requirements outlined above. Additionally, it is accepted, without controversy, as a distinguishing factor in the classification of object-oriented interfaces. Indeed, it is often the most remarked upon attribute of object-oriented interfaces. As a dimension for classification, it has the additional advantage that, the determination of a particular interface's position along the dimension is unambiguous. Judgments concerning the required command sequence of an interface can be made straightforwardly without relying on subjective impressions.

The interesting result from the observational study: that TopView subjects (non-object-oriented group) preferred to use an object => action syntax, strongly suggests that this dimension has consequences for user performance.

Furthermore, an object => action syntax usually (i.e. in a non command line style interface) entails that objects are presented first, so investigation of the use of syntax could be interesting in respect of other factors.

3.1 Syntax: the Findings of the Observational Study and a Rudimentary User Model.

As mentioned above, in the observational study, subjects using a non-object-oriented interface (TopView) were found to prefer to use an object => action syntax in 60.75% of instances, despite a number of dissuasive factors (see Chapter 4). Additionally, there were no instances where the subjects using an object-oriented interface (Lisa) attempted to use an illegal action => object command sequence. In general terms this finding can be expressed as, 'an object => action syntax is preferred for the computer supported performance of common office tasks', where, it is assumed that ease of use can be inferred from the preference.

This finding can be combined with the notions of the User Structure for user modelling (expressed in Chapter 2) in order to derive a rudimentary User Model. It is expected that the results of the
subsequent empirical activity can then be used to modify and enhance this User Model as a better understanding of syntax usage is gained.

It was proposed in Chapter 2, that the User Structure should specify the processes and representations of the user, in general terms. Particular instances of user behaviour during task performance can then be modelled using the structure. Here, a User Structure will be derived which concerns the representations held and the processes performed by the user during the observational study. The structure will focus on the task element of interest, the use of syntax.

The User Structure assumes that when performing a task using a given device, the user undertakes two primary activities: assimilation of task instructions and implementation of task goals. To achieve these activities the user must hold a number of successive representations of the task at different levels of description. Each representation is transformed by a specified process into the next representation, which may be expressed at the same or a different level of description.

The rudimentary User Structure, see Figure 5.1, attempts to specify the processes and representations pertinent to syntax usage.
Figure 5.1
Rudimentary User Structure
First, the user (of either device) works towards the formulation of task goals in accordance with the task instructions. Due to the artificial experimental setting for the study it is assumed that the subjects did not have any predetermined goals with respect to the tasks or task materials. The processes and representations of the right hand side of the structure are provided as an illustration of a possible method of deriving task goals, i.e. that the text instructions undergo a number of transformations ranging from the perception of single letters to the specification of sentence meaning and the abstraction of task goals. The processes and representations shown here are not intended to be a complete or a correct model of human information processing. It is recognised that many intermediate representations may be held.

The task goals are expressed as an object action pair. The object and action are unordered initially, as denoted by the curly brackets, \{\}. The pair are then ordered with the object assuming the primary position. As yet, there is no explanation for why the object should be the first element of the task goal expression.

Following specification of the task goals, the user begins their implementation. The working representation of the task is re-expressed at successively lower levels of description until a specification for physical action is obtained. The two notations for objects and actions (e.g. object and object ') are used to indicate that the same referent is intended in each case, but that the form of expression for it may differ. For example, object refers to memo (general), whereas object ' refers to memo-87 (specific). It is assumed that during this phase the user is aware of the device which is to be used and that s/he has a knowledge of the requirements for interaction with that device e.g. knowledge of the command words available; knowledge of the required syntax; knowledge of the method of input etc. As this user structure is intended to focus on syntax usage, only the device features of interest will be specified (i.e. the required syntax and the method of input technique).

The ease with which the task goals are achieved can be assumed to depend on the degree of compatibility between the current representation held by the user and the way in which the device requires the task elements to be represented.

The User Structure is applied to provide an example of a Rudimentary User Model, see Figure 5.2. This model is an instantiation of the structure as it describes a particular instance of user behaviour, as found in the observational study. The model describes the behaviour of a TopView subject, using an object => action syntax. This syntax is not strictly correct, but the device would accept it without providing negative feedback. The other device feature of interest is the method of input, which in the case of TopView is selection by mouse.
Figure 5.2
Rudimentary User Model (TopView Subject)
The model, depicted above, describes the behaviour of a TopView subject performing the first task. In the model an attempt is made to describe the actual content of the representations. The representation of concepts is not fully shown, but it is expected that they could take any appropriate form e.g. a propositional representation. The representation for motor programs has been abbreviated such that a mouse movement is denoted by MM and a mouse click is denoted as MC.

In terms of this model, TopView users would tend to avoid using an action => object syntax as they would have to perform an additional process on their working representation to transform it to the re-ordered expression, 'ERASE, MEMO-87', at the point marked •. It is assumed that to enhance ease-of-use, TopView users have adopted an interaction strategy (i.e. use of object-action syntax) which is more closely compatible with their representation of the task.

4. Method of Experimentation

The previous section has argued the case for the empirical activity focussing on a single dimension of object-orientatedness, syntax at the Communications Level. A strategy for experimentation must now be decided. A number of decisions must be made to determine an appropriate strategy.

The first decision concerns whether real or simulated interfaces should be employed for experimentation. Real interfaces have the advantage of maintaining external validity, but, as has been mentioned previously, their major drawback is the presence of confounding variables. Interfaces which differ in their command syntax (i.e. object => action versus action => object) are likely to differ in a number of other respects which may have unspecified effects on user performance. By contrast, the use of simulated interfaces will allow a greater degree of control to be exerted over variables since alternative interfaces can be designed which differ only in their syntax. This strategy is to be preferred. Here, simulation refers to a device which reproduces the behaviour of the system under investigation. It achieves this by representing, in a controlled context, those system entities which are mutually influential when the target system operates to fulfil its function (see Life, in press).

Next, the nature of the interface simulation must be decided. Two alternatives will be considered, paper-based simulation and machine-based simulation. Paper-based simulations have the advantage of being relatively economical (in terms of time and resources) to prepare. An interface might be simulated in this way by having the dialogue appear on cards. To interact, the user reads the card displaying system prompts and responds by selecting the appropriate card for a response. (For an example of the use of this approach see the MacAll project, Smith et al. 1987).
It is obvious that, while useful for investigating such topics as the feasibility of dialogues, the paper-based simulation affords the user a qualitatively different experience from that of interacting with a computer interface. On the other hand, a machine-based simulation, while costly to prepare in terms of programming time and machine resources, has the potential to offer the user an experience of a similar context to that of using an extant interface. This higher degree of external validity is to be preferred here.

The final decision concerns which machine and language combination is to be used to implement the simulation. The basic requirements for the machine simulation are that:

i) it is quick and easy to implement;
ii) a relatively small effort is needed in terms of time spent programming;
iii) it has a realistic visual display (important for encouraging users to believe that they are interacting with a real interface).

The machine and language combination which was found to meet these requirements best was HyperCard which runs on an Apple Macintosh.

In the following sections, HyperCard and its programming language will be described and the reasons why it is particularly good for interface simulation will be outlined.

4.1 Introduction to HyperCard

HyperCard is a software tool, developed by Atkinson. It is difficult to describe because it can exist in a number of guises and it can be used at a number of different levels. In an interview with Atkinson, Goodman (1987) reports that, "HyperCard is an authoring tool and an information organizer. You can use it to create stacks of information to share with other people or to read stacks of information made by other people. So it is both an authoring tool and a cassette player for information". The design of HyperCard is derived from a combination of two information handling techniques: hypertext and the card index. Hypertext has been described as, "a method of presenting information on a video screen by providing intuitive links between related elements", Whitby (1987). In other words, if a screen is displaying a page of text the user might be able to click on key words or phrases, using the mouse, which would result in a different display which gives further details concerning the particular word or phrase.

HyperCard has five levels of usage and these are responsible for determining whether HyperCard is used actively or passively. The most basic level of usage is the Browsing Level while the most advanced is the Scripting Level. At the Browsing Level, a user can only look at
cards and is prevented from altering their contents, whereas at the Scripting Level the user is free to invent stacks; creating new cards, fields, buttons, artwork etc. Between the two extremes, there are levels which allow text editing, manipulation of artwork and simple creation and modification of buttons and fields.

HyperCard is based on a hierarchy of objects. The most basic objects are stacks, cards, buttons and fields. Central to HyperCard is the card object; the screen space is always occupied by a card when HyperCard is running. The card can contain a number of different sorts of information. It can contain static information such as a graphical display (e.g. a map of Europe) or a piece of text (e.g. a description of a book). It can also contain other types of object e.g. buttons and fields. Buttons define screen areas which, when clicked with the mouse, can make something happen (e.g. a button may enable a different card to be viewed) and fields define text areas which can contain static text or text input by the user.

Examples of buttons, fields and a card are shown in Figure 5.3.
The card is made up of two separate layers, one layer is unique to that particular card and the other layer, the background, is shared by related cards. Both layers can contain buttons, fields and artwork etc. and the contents of both are usually visible on the screen, simultaneously.

HyperCard offers a rich graphics environment. Pictures can be created directly using the many Paint tools or pictures and parts of pictures can be copied between cards.

A group of related cards form a stack. Stacks can contain (almost) any number of cards. A
particular stack is usually designed to provide some sort of function and so can correspond with the notion of an application. For example, a stack might offer a diary facility with a different card for each week or it might be a tutorial for learning about European Geography.

The dynamics of the system are provided by a mechanism of message passing and inheritance between objects. The system constantly monitors "events" (e.g. a mouse click; depression of the Shiftkey; entry of specific command words etc.) and on detection, sends a message which passes through the hierarchy of objects. First, the message is sent to the lower level objects (e.g. buttons), if a button's script contains an appropriate event handler then the script is executed, otherwise the message passes up to the objects of the next level of the hierarchy e.g. cards. This inheritance mechanism is similar to that found in Smalltalk.

The author of a stack specifies the particular dynamics of that stack by writing in the scripts that handle events. Scripts are written in HyperTalk, the special-purpose high level programming language of HyperCard. Any of the objects (cards; stacks; buttons; fields etc.) can have a script associated with it. As described above, the script determines the consequence of a particular event. A simple button script may specify that the computer emits two beeps when the mouse is placed within it and the mouse-button is clicked.

The script may look like this:

```
on mouseUp
  beep 2
end mouseUp
```

A more complex script may specify that when a button is pressed the content of a particular field (called "how many") is examined. If the number contained there is greater than 2, then a message appears on the screen reading, "This number is too big!" (the dialogue box containing this message will have an "OK" button, which, when activated, causes the box to disappear) and the field is emptied. If the number is 2 or less then a new card (with the ID number 2456) will be shown on the screen. On moving to this card, the present card will be overwritten in a manner which makes it appear as if the card is being wiped over from right to left (command = visual effect wipe left).
The script would look like this:

```hypercard
on mouseUp
    get card field "how many"
    if it > 2 then
        answer "This number is too big!"
        put empty into card field "how many"
    else
        visual effect wipe left
        go card ID 2456
    end if
end mouseUp
```

At whichever level it is used, HyperCard offers an interesting and novel environment and as the complexity of the UserLevel is increased the flexibility and scope for creativity multiply.

4.2 The Use of HyperCard to Simulate Interfaces

In this section, the particular features of HyperCard that are exploited for interface simulation will be described.

HyperCard is a powerful tool for interface simulation, when used at the scripting level, because most aspects of the screen display and the user input can be controlled by the stack author. A high level of control is achievable since the programming language, HyperTalk, is relatively easy to learn for anyone with minimal programming experience. Any programming is implemented immediately and so its effects can be adjusted rapidly - this is not true of environments where compiling is necessary.

The cards' structure is particularly suited to the development of interface simulations as different stages of the dialogue can be represented on different cards. This strategy obviates the need for programming full functionality without the absence becoming apparent. Additionally, there is potential for building support for the experimental session e.g. presentation of task materials; data logging.

The high quality of the graphics display and the way in which user input can be tracked from either the mouse or the keyboard, enhance the range of interfaces which can be developed.

For experimental purposes, a User Level can be set which prohibits subjects from being exposed to the full range of HyperCard functionality, so the interfaces developed are rendered fairly robust.
4.3 General Description of Interfaces Generated Using HyperCard

The interfaces which were developed to support the empirical work reported in this thesis had a common structure. Their structure will be described.

All the interfaces comprised three elements:

i) one card displaying the initial 'option' screen;
ii) a set of cards representing the main body of the interface functionality;
iii) a few cards for recording interaction data (and other types of experimental support).

The card for the initial screen presented the task elements (objects and actions) to the user and allowed the user to select the items required. The script of this card specified most of the control structures for the interface. The display on this card typically differed between the various versions of an interface e.g. for one group the objects might appear at the top of the display and the actions below, while another group might be presented with the reverse arrangement; one group might have to enter a command sequence which had a different syntax from that required of another group.

The set of cards which made up the main body of the interface was the same for all groups. There were cards for each object in the scenario. In this set of cards, 'local' messages were specified e.g. if the user had elected to send an item then a dialogue box would appear on the card of that item requesting details for transmission, the card script would specify that there was a pause in the interaction (using the 'Wait' command), before displaying a message that the object had been sent. This arrangement was found to be very convincing for most subjects; some even asked if the system was available as an off-the-shelf application.

A number of cards were set aside to receive data e.g. times of button presses and keyboard entries and recordings of which buttons were pressed etc. In general, these cards contained a single scrolling field (i.e. unspecified length) and items were simply assigned to the next empty line of the field. The mechanism was controlled by the initial card. The subjects never saw these cards and the data were transferred after each session. Additionally, for the first experiment, there were a number of cards which displayed the task instructions and these were presented to the subjects at the beginning of each task.
5. Concluding Remarks

This chapter has argued for an empirical approach which investigates particular device features in isolation. A dimension has been selected to become the focus for future experimentation and a method for investigating this dimension has been outlined. The empirical work which is based on this approach will be reported in Chapters 6, 7 and 9. In addition, a rudimentary user model based on the results of the observational study has been derived. It is expected that the future empirical work will modify and enhance this model.
Chapter 6
Investigating the Usability of an Object First Syntax under Different I/O Level Conditions: the First Experimental Study (E1)

1. Introduction

This chapter implements the empirical approach outlined in Chapter 5 to investigate the conditions which influence syntax usage.

An experiment is designed which aims to explore the effects of factors at the I/O Level on the usability of particular syntax structures (at the Communications Level). It contrasts the use of two input techniques: a command line style technique and a Pick & Point technique, over the two syntax structures: object => action and action => object.

The experiment relies on the development of interface simulations in order to control both the variables believed to influence syntax usage and those spurious to the undertaking. A description of the simulated interface and its different versions is provided.

The design and method of the study are described. An experimental hypothesis is formulated which reflects the findings of the observational study.

The results are analysed and discussed in terms of the experimental hypothesis. Following this discussion, the findings are used to further the analytic activity by modifying the User Model. The findings are also expressed in a summary table which is based on the levels structure proposed in the framework.

Finally, some possible future directions for the empirical activity are considered.

2. Aims of the Study

The primary aim of the experiment is to investigate the usability of different syntax structures at the Communications Level (i.e. command words; selection of items from menus etc.). In particular, two syntactic structures are compared; an object => action sequence and an action => object sequence.

There are two reasons why syntax is of interest:
i) it has been proposed as an important dimension for the classification of interfaces with respect to their degree of object-orientedness.

ii) the results from an observational study showed that subjects preferred to use an object => action syntax in 60.75% of instances of syntax usage, despite there being a number of factors to dissuade them from doing so.

It has been suggested that the command syntax of an interface might contribute to its ease-of-use. Syntax alone could be a major determinant of an interface's usability or, alternatively, it might be syntax in combination with other factors that is responsible. In other words, particular configurations of syntax and other factors might be easy to use while other combinations might cause problems for the user. Some possible factors which might interact with syntax are:

i) the type of task - the particular semantics of the objects and actions involved in a task might have an influence as might the task structure i.e. the order in which operations/sub-tasks are performed.

ii) the method of input - the way in which the user interacts with the device e.g. whether a command line style or a mouse/Pick & Point style is used.

iii) ratio of object types to action types e.g. a particular syntax might be favoured if the most prevalent component came first in the sequence.

iv) the task context - the most usable syntax could depend on the nature of the tasks in a particular domain. e.g. one domain might feature tasks which focus around objects whereas another might have tasks which are made up of sequences of actions.

In the observational study, TopView subjects seemed to prefer to use an object => action syntax, in the majority of instances. From this preference it was inferred that an object => action syntax was easier to use. It was also noted that the Lisa subjects had no problems adhering to an object => action syntax. Due to the particular interfaces selected, this object first advantage was only observed under conditions of Pick and Point style input, the style of input which is commonly associated with direct manipulation interfaces. It could be the case that the effect is restricted to this style of input. The experiment reported in this chapter has two aims:

a) to discover whether there is an object first effect, under controlled conditions;

b) to find out whether the object first effect is observed under different conditions of input style, Pick and Point versus command line.

As explained in Chapter 5, the use of extant systems to explore the effects of particular variables is inappropriate due to the presence of confounding variables, so a simulated interface will be developed.
3. Background

Syntax at the Communications Level is an important issue in the design of interactive dialogues between the computer and user. A number of previous studies have focussed upon syntax and have attempted to establish the conditions under which particular sequences are preferred. Clearly such an endeavour has a role in influencing system design of the future.

A number of studies performed by Barnard and colleagues have focussed on the structure and content of interactive dialogues. Those concerned with structure are of particular interest here.

One study investigated the ordering of arguments within a command sequence (Barnard, Hammond, Morton and Long, 1981). The application that was investigated was a message decoding system in which each command had two arguments; one of these arguments would be the direct object and the other the indirect object of the command e.g.

```
ERASE DIGIT 9 MESSAGE 7
```

This command would cause digit 9 from message 7 to be erased.

Three factors concerning syntax were of interest:

i) the relationship between the argument order and the order used in natural English.

ii) the consistent positioning of the recurrent argument.

iii) the relationship between argument order and the position of argument values on a VDU.

In the interactive, on-line version of the task it was found that positionally consistent systems were the most readily learned, but only if the recurrent argument was found in the first position. An effect due to the position of the direct object (i.e. that it should be placed so as to be compatible with natural English syntax) was found for the positionally inconsistent systems, but then only for certain commands. On average, the positionally inconsistent group having an "unnatural" structure performed no worse than the group which had a "natural" structure. This study illustrates that the commonly held assumption that device syntax should adhere to the principles of natural language is not necessarily founded.

Another study by Barnard and Hammond (1982) investigated syntax, but here syntax between command and its argument. A menu-driven data-base application was studied. The system had five functions and five objects. The data-base could be interrogated by selecting a command sequence consisting of one of the functions and one of the objects. The positioning of
the items in the menus was balanced across subjects (either the function set or the object set appeared as the top half of the menu). Subjects received instructions which were balanced such that half mentioned the object first and half the function first. Prior to the experimental session subjects were briefed about the system; half the subjects were told about the system's functions and half about the system's objects.

Two versions of the experiment were run. In the first version, the subjects were allowed to use 'free' syntax i.e. they could choose whether to specify the object or the function first. Here an overall bias was found towards a dialogue in which the function is selected before the object. This bias was modified by the type of briefing the subjects had received (i.e. subjects who had been told about the objects were more likely to select the object first). There was also a tendency for subjects to pick menu items from the top half of the display first and to choose the item mentioned first in the instructions as the first item in the command sequence.

In the second version of the experiment the required syntax was fixed, half the subjects had to specify the object first and the other half were required to specify the function first. Under these conditions it was found that the initial briefing and the structure of the questions had less marked effects on performance. Also there was no performance difference between the two syntax groups. For the 'fixed' groups compatibility with natural language syntax does not seem to be important.

A comparison between the two groups, 'fixed-' and 'free-' syntax, revealed that there was a difference in the learning of the system for the two groups. Subjects in the 'free-syntax' group seemed to acquire less knowledge about the system: they requested 'Help' more often in the early stages of use and experienced more difficulty in transferring from simple to complex problems.

The differing results for the two versions of the experiment were explained in terms of the differential task demands imposed by the two variants of the system.

A further study concerning syntax was reported by Cherry (1986). Subjects were required to use one of two command languages designed for a text editor. The two languages differed in their syntax; one being object-action and the other action-object. (NOTE: due to the application the objects were numbers which referred to the line number/word number etc. e.g. 'insert 25'; 'replace 5, 7'). Contrary to intuitive expectations, there were no differences in performance between the two syntax groups on any of the measures taken (percentage of tasks completed; percentage of erroneous commands; editing efficiency).

These latter two studies show that in one menu system and one command-line system there was no effect due to syntax (o-a vs. a-o). However, the preference for an action-object syntax
in the 'free' syntax condition of the Barnard and Hammond study indicates that the two orderings are not always found to be equivalent and factors, such as cognitive load imposed by the task, might be responsible. Clearly the relative usability of the two syntax structures over two input styles cannot be assessed from these previous studies which are set in different application domains.

It has been suggested in a paper that describes a kit for generating interfaces (Lieberman, 1985) that a good principle for interface design is to provide the user with both options (object-action and action-object). The kit (EZWIn) allows this type of interface to be developed. However, the consequences of this approach may not be favourable as the Barnard study showed a learning deficit in the free syntax condition. A trade-off between time to learn and ease-of-use once learned, might have to be considered.

Given the above studies on syntax, it would seem that there is little empirical evidence to support the notion that an object => action syntax is easier to use. However, it is also true that they did not demonstrate a clear advantage for action => object syntax. So the preference for object => action syntax found in the observational study cannot be explained with reference to these previous studies. It is interesting to note that many new commercial systems have followed the lead of the Xerox Star and have incorporated an object => action syntax; deeming it to be more 'natural' to use. Clearly there is a need for further empirical investigation into the use of syntax to determine the conditions under which a particular sequence is advantageous.

4. Details of the Simulated Interface

The simulated interface was developed to support the comparison of the use of two syntax structures over more than one style of input, keeping other factors, which might influence syntax usage, under control.

4.1 Requirements for the Interface

On the basis of the above, the general requirements for the experimental interface were determined. These are that:

i) the versions generated from the interface should vary only with respect to their syntax and input technique.

ii) the mode of representation of the entities should be constant: also it was thought best to restrict representation to a textual representation (rather then an iconic one). As object-oriented and graphical interfaces are assumed synonymous by a number of authors this
was also an exercise in investigating the notion of object-orientedness outside of the context of graphical and/or Direct Manipulation interfaces.

iii) The ratio between the number of actions which are paired with the objects (and vice versa) in the system should be held constant.

Overall the variations to be compared should have the same TOTAL number of actions and objects and the pairings should yield identical numbers of options per object (or action).

E.g.

Total number of objects = 4 { A, B, C, D}

Total number of actions = 4 { 1, 2, 3, 4}

Number of options per object (or action) = 3

<table>
<thead>
<tr>
<th>OBJECT VERSION</th>
<th>ACTION VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>A 2</td>
<td>C</td>
</tr>
<tr>
<td>3</td>
<td>D</td>
</tr>
<tr>
<td>B 2</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>D</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
</tr>
<tr>
<td>C 4</td>
<td>A</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>1</td>
<td>B</td>
</tr>
<tr>
<td>D 1</td>
<td>B</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
</tr>
</tbody>
</table>

This organisation contrasts with the situation that often occurs with extant systems where different objects may have different numbers of actions associated with them - a document may be associated with six or more actions; some of which may be unique to documents (e.g. Proofread) whereas a clock may only have one or two actions associated with it. The result is unequal option sets between the two versions (e.g. 6 : 1 becomes 1 : 6). Number of options is a potential confounding factor because the preferred syntax might be correlated with the most (or least) frequent option type.
4.2 Description of the Simulation

As described in Chapter 5, the simulated interface was written in HyperTalk, the
development language of HyperCard, on an Apple Macintosh II. It was a simulation of an
interface in the sense that the system did not have full functionality, but apparent
functionality was suggested to the user by the provision of messages. As an illustration, if
asked to send a memo the user would type in the name of the intended recipient and after a
pause the system would indicate that the memo had been sent when, in fact, it had not. Also,
when changes were made to a display such as in 'sorting the directory'; another card would
be presented to the user which showed the directory in its new state, rather than any change
taking place on the original view of the directory.

There were a number of versions of the interface and the differences between the versions
will be described, first a description of the essential features of the interface will be given.

All versions of the system had six objects (memo; calendar; graph; mailbox; report;
directory) and six actions (send; edit; copy; display; print; sort). Each object (or action) could
be paired with three actions (or objects). Three of the objects could have their contents edited
by the user (memo; report and graph) and three were static information sources (mailbox;
calendar and directory), this difference being signalled by having the different groups
paired with different action sets.

There was only one instance of any object (i.e. there was one memo, one graph etc.), an
arrangement chosen to simplify the syntax to its most basic: object => action or action =>
object and to preserve exactly, the object:action ratios.

The interface also included facilities to support the test session. The task instructions were
presented on screen. Subjects had only one opportunity to perform each task, after the task
had been attempted, irrespective of the outcome, the next task description was presented. On
the cards containing a task description there was an 'OK' button; subjects were instructed to
click over this (using the mouse) when they had read the description. The initial screen of
the interface was then presented (either with the Pick & Point presentation or the
Command-line presentation).

During task performance the system logged the commands chosen (Pick & Point group) or the
strings typed (Command groups) as well as timing information (time to read the description;
time to select the first and second commands for the Pick & Point condition).

The versions of the interface differed in three respects:
i) syntax (object => action versus action => object);
ii) input style (command line versus Pick & Point style);
These differences were only apparent in the first screen shown to the user; once the user had specified the command, the system continued in the same way for all groups.
iii) task descriptions - the ordering and wording of the task instructions was balanced across the subjects.

In the Pick & Point condition, the initial screen displayed six boxes which were labelled either with six object names, or with six action names. To execute a command sequence (consisting of an object-action pair), the user was required to click on the appropriate box, using the mouse input device. This action caused a pop-up menu to appear giving the user three options (either all actions or all objects). The user then had to select one of the options to complete the command sequence.

An illustration of the initial screen for the Pick & Point/object => action Condition [OP] is given in Figure 6.1.

![Initial screen display for Pick & Point interface](image)

Figure 6.1
Initial screen display for Pick & Point interface

In the Command-line condition, the initial screen displayed a command-line box with a flashing cursor. The user was required to type in the two words of the command sequence and
then to press the 'Enter' button on the keyboard. If the typed sequence was incorrect for any reason (e.g. the words were in the wrong order; a word was mis-spelt etc.) a message box appeared which prompted the user to try again.

An illustration of the initial screen for the Command/action => object Condition [AC] is given in Figure 6.2.

Please type in a command:

(Press Enter)

Figure 6.2
Initial screen for the command line interface

An example of an error situation appears in Figure 6.3.

Please type in a command:

edit reprot

(Press Enter)

Incorrect Command -
Please Try Again

OK

Figure 6.3
Error message for Command Line interface
5. Method

5.1 Design

A 2 x 2 between subjects design was adopted where the independent variables were: syntax (object => action; action => object) and method of input (typed command line; selection from options). This yielded the four groups:

```
SYNTAX

<table>
<thead>
<tr>
<th>INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>STYLE</td>
</tr>
</tbody>
</table>
| \
| Command line |
| | object => action |
| | action => object |
| \
| Pick & Point |
| | object => action |
| | action => object |
| \
| AC |
| OC |
| AP |
| OP |
```

The two dependent variables which were to be recorded were errors (in command usage) and time taken to select commands (Pick and Point groups only). It was expected that the timing data for the Pick and Point groups would be analysed in the event of similar error rates. It would be expected that a faster time to select commands would indicate that lower effort was required and so greater usability could be inferred. It was decided not to analyse timing data for the Command group, because the results would not be particularly meaningful. The time taken to begin typing the command does not necessarily indicate the time taken to select the command word, it might instead indicate how well subjects could spell; or their familiarity with a QWERTY keyboard (i.e. subjects who were very familiar with a keyboard would have faster times than those who were not familiar).

Two factors were balanced across subjects these were:

i) the sequence of presentation of the tasks;

ii) the wording of the instructions (whether the object or the action appeared first in the instruction e.g. compare 'Edit the report' with 'The report should be edited').
5.2 Subjects

Thirty-two subjects were tested. Their participation in the study was voluntary. Subjects were recruited from the Bedford Way Building (departments of Psychology; Ergonomics and Geography) and the adjoining Institute of Education.

The requirements for subjects were that they should:

- have some general computer experience
- be familiar with office tasks
- be educated to degree standard

Twenty female subjects and twelve male subjects participated in the study. The mean age of the subjects was 27.6 years with a range of 20-39 years. The majority of the subjects had some previous experience mostly with statistics, games or word processing. All were naive to the simulation. Subjects were assigned randomly to conditions.

5.3 Tasks

The tasks were drawn from the domain of common office tasks (i.e. those that are common to office professionals irrespective of their discipline). Thus, the task context was held constant for all subjects. Each subject had eighteen tasks to perform. These resulted from pairing each object with each of its three actions \([6 \times 3]\) (or vice versa), thus ensuring equal exposure to all of the objects and actions. Also possible effects of task structure were eliminated by having each task accomplished by performance of a single sub-task (no ordering of sub-tasks was necessary).

The instructions were simple and literal i.e. they contained the actual command words e.g. 'Sort the directory by number'. This style was adopted to ensure that the time data did not contain a large/unspecified interpretation component (such as would be expected with more loosely described instructions).

See Appendix G for list of tasks.

5.4 Training

All subjects were trained using purpose-written training materials (also written in HyperCard). These were designed to introduce subjects to the various interaction techniques they would be expected to use in the test session (e.g. use of the mouse; editing techniques etc.). The command-line groups (AC and OC) had one additional screen of training material.
which demonstrated how to enter command strings.

5.5 Experimental Session

The experimental session for each subject followed the same schedule:

- General Instructions
- Training
- Specific Instructions (about Test Session)
- Test Session
- Questionnaire & debriefing (the questionnaire was intended to ascertain personal details concerning the subjects' ages, sex, previous experience etc.)

The session lasted approximately 30 minutes in total.
See Appendix F for the instructions which were given to subjects.

During the Test Session, the experimenter compared performance against a checklist consisting of a listing of the correct command sequences for each task.

A video recording of each performance was made. The camera was focussed on the screen such that all the subjects' interactions were recorded. No sound recording was made.

The simulation automatically logged the commands chosen and the time taken to select each command for the Pick & Point groups.

The experimental hypothesis for this study was based on from the results of the Observational study and it asserts that:

\( \text{an object} \Rightarrow \text{action syntax should be easier to use in all instances} \)
\( (\text{where ease of use is inferred from a tendency towards error-free performance}). \)

6. Results

6.1 Errors

The number of errors in specifying the command sequence was recorded for each subject. The error data were analysed using non-parametric tests for significance (Mann Whitney U-test
for the difference between means and Kruskal Wallis Analysis of Variance) because the low incidence of errors resulted in the data having a skewed distribution.

6.1.1 Overall Number of Errors

Table 6.1 shows the overall number of errors made by each group (spelling errors and spacing errors made by the command groups have been discounted). The totals include errors made in attempting to recover from an initial error.

<table>
<thead>
<tr>
<th>Group</th>
<th>No. of Errors</th>
<th>Mean</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>OC</td>
<td>29</td>
<td>3.625</td>
<td>1.996</td>
</tr>
<tr>
<td>OP</td>
<td>11</td>
<td>1.375</td>
<td>1.576</td>
</tr>
<tr>
<td>AC</td>
<td>7</td>
<td>0.875</td>
<td>1.053</td>
</tr>
<tr>
<td>AP</td>
<td>11</td>
<td>1.375</td>
<td>0.857</td>
</tr>
</tbody>
</table>

Table 6.1
Overall number of errors by group

The graph, shown in Figure 6.5, depicts the mean number of errors made by each group. The graph shows an interaction between the two independent variables, syntax and input style.
As can be seen the hypothesis that an object => action syntax is easier to use was not supported. For the Command groups, it was found that OC made at least four times as many errors as AC, while there was no difference between the Pick & Point groups.

The difference between groups was found to be significant using the non-parametric Kruskal-Wallis Analysis of Variance (H = 8.293, p < 0.05 [Corrected for ties, H = 8.727]).

Figure 6.5
Graph showing mean number of errors made (by group)
6.1.2 Classification of Error Types

The errors were then divided into five main types.

I Wrong Action - where an action which does not conform to the task instructions is selected.
E.g. Subject selects 'Edit' when told to 'Display the calendar'.

II Wrong Object - where an object which does not conform to the task instructions is selected.
E.g. Subject selects 'Report' when told to 'Copy the memo'.

III Substituted Action - where an action that is not included in the command set is used.
E.g. Subject types 'Change memo' for 'Edit the memo'.

IV Substituted Object - where an object that is not included in the command set is used.
E.g. Subject types 'Sort meetings' for 'Sort the calendar'.

V Syntax - where the inappropriate syntax is used.
E.g. if the subject is in an object => action group s/he might type 'Print directory' instead of 'Directory print'.

Errors of types III, IV and V can only be made by the Command groups.

Table 6.2 shows the number of errors of each type that were made by the four groups.
There were no significant differences between the groups for the error types I and II (Kruskal-Wallis test).

Also there were no significant differences, between groups AC and OC, for the error types III and IV (Mann-Whitney U-tests: U=20; U=25.5, p>0.05).

There was a highly significant difference between AC and OC for error type V (Mann-Whitney U-test: U =3, p<0.001).

### 6.1.3 Errors and Trial Type

As the wording of the instructions was balanced so that half the instructions for each subject had the object appear first and half had the action appear first, some trials would be
congruent (i.e. required syntax is in same order as instruction) and some would be incongruent (i.e. syntax is not in same order as instructions). It was decided to check whether the occurrence of an error was related to the congruence of a trial. It is plausible that more errors occur on incongruent trials.

The following table shows the number of errors of each type which were made on congruent and incongruent trials.

<table>
<thead>
<tr>
<th>Error type</th>
<th>Trial type</th>
<th>Incongruent</th>
<th>Congruent</th>
</tr>
</thead>
<tbody>
<tr>
<td>I Wrong Action</td>
<td>11</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>II Wrong Object</td>
<td>10</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>III Substituted Action</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>IV Substituted Object</td>
<td>6</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>V Syntax</td>
<td>6</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td><strong>TOTALS</strong></td>
<td><strong>35</strong></td>
<td><strong>23</strong></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.3
Number of errors made on congruent and incongruent trials

Evidently there are more errors made on incongruent trials in total; however when the individual figures are examined by error-type, the difference is not significant. It is particularly interesting to note that the type of trial did not seem to have an effect on syntax errors.

6.2 Time Data

Data concerning the time taken (precision 1/60th second) to select the first command and the second command (AP and OP) were recorded in order that if there were no differences in the
error data then differences in terms of time could be explored. (A sequence would be considered the easier to use if it led to faster selection times than another, accuracy being equal).

As no differences were found between the Pick & Point groups in terms of their errors, the time data for the two groups were compared.

Parametric t-tests and Mann-Whitney U-tests were performed on the selection time for the first command for each task. Using the t-test, only the selection time for Task 17 was found to be significantly different for the two groups (AP and OP) in the direction OP faster than AP (t=2.0295, p<0.05). The non-parametric test showed significant differences between the times for tasks 4 (U=15, p<0.05), 12 (U=12, p<0.05), 13 (U=15, p<0.05) and (17 U=3.5, p<0.01) in the direction OP faster than AP with the exception of Task 4.

This result is very scant evidence for an object => action advantage since there was a significant difference on only 1/6th of the tasks (3 out of 18).

An analysis of variance was also performed on the data. This revealed a significant difference between the tasks (F (17,126) = 3.0809, p<0.01), but no significant differences between the two groups (F(1,126) = 0.3221, p>0.05) and no interaction effect (F(17, 126) = 0.454, p>0.05). The difference between the tasks is not surprising as subjects are expected to improve in the rapidity of selection as they became more familiar with the system and the experimental set-up in general.

The times for selecting a second command showed no difference between the groups, p>0.05.

7. Discussion

In this section, the results of the experimental study will be discussed.

First, their implications will be discussed in terms of the experimental hypothesis and the results of the previous study. The results have potential for informing the rudimentary User Model, outlined in Chapter 5, and an enhanced version of this will be described. Finally, the results were found to be useful for beginning to document the conditions which influence syntax usage. This information is expressed in a Summary Table of Results.
7.1 Implications of the Results

It is clear from the analysis of both the error data and the time data that the experimental hypothesis was not supported. Neither of the object => action syntax groups (OC and OP) performed better than the action => object groups (AC and AP). Group OC made significantly more errors than any of the other groups (p < 0.05) and this group also made significantly more syntax errors than the AC group (p < 0.001). No difference in performance was found between the two Pick & Point groups (on error or time measures).

Two aspects of these findings merit further discussion:

i) why the OC group made more errors than the AC group;
ii) why the results from this study differ from the syntax results of the observational study.

7.1.1 On the Difference in Performance found between the OC and the AC Groups

Contrary to expectations based on the experimental hypothesis, the object => action group made more errors (particularly syntax errors) than the action => object group.

This result could be explained in terms of interference from existing knowledge about natural language syntax. Subjects in the Command groups were required to type in a two word phrase consisting of an object word and an action word. For the OC group this command sequence is contrary to natural English syntax for the imperative.

E.g.

\[
\text{directory} \Rightarrow \text{print}
\]

instead of:

\[
\text{print (the)} \Rightarrow \text{directory}
\]

Therefore, one might expect that there would be some interference from the subjects' existing knowledge of natural English which would lead to a higher error production (particularly of syntax errors) than for a group having a command sequence consistent with English (such as AC). If this were the case then one might expect an o => a syntax to suffer more interference in all instances, i.e. irrespective of input method, but this was shown not to be the case (OP...
did not produce more errors than AP).

It could be argued that it is precisely the salient difference between the two object => action groups, i.e. the requirement for text input for Group OC, which activates the interfering factor. In typing, the subjects are expected to draw upon some relevant sources of existing knowledge in order to specify the command sequence e.g. the spelling of a number of common words; the semantics of some common words. They may also draw upon knowledge about syntax (production of English normally requires such knowledge; as in writing or typing) The result would be a facilitative effect for the AC group but an interfering effect for the OC group.

Yet for the Pick & Point style conditions, the subjects are not required to call on specific knowledge about the production of English as they do not have to type in words, so they may not access the mechanisms responsible for syntactic processing. However, instead, they may draw upon more generalized knowledge about language and particularly English language to recognise the various options.

This result is informative for the development of the User Model and it gives an insight into the conditions which influence syntax usage.

7.1.2 On the Discrepancy in Findings between this study and the Observational Study.

As noted previously in this section, the results of this study are not consonant with those of the Observational study (reported in Chapter 4) because an object => action syntax was not found to be preferred (where preference is indicated by a low incidence of error and a rapid selection time). A good starting point for exploring this discrepancy is to consider differences in experimental conditions. Essentially, there are three differences, each of which might be responsible for the discrepancy. These are discussed below:

i) Differences in the computer systems tested;
It could be that the factors responsible for the object => action preference were controlled in the experimental study and so their effect was minimized. An example of a possible factor is the object-action ratios. In the experimental study there were exactly similar object => action ratios between groups.

ii) Differences in the instructions given to subjects and the type of task;
In this experimental study, the instructions were of a literal, verbal form i.e. subjects were told 'x the y' or 'the y should be x-ed' with 'x' and 'y' being the actual command words. It
could be that this form of instruction results in different behaviours than that observed in the observational study. This study required subjects to translate from an exact instruction to a device implementation of the instruction; whereas the observational study examined the interpretation of a device-independent task description and its implementation on a particular device.

One possible explanation for an effect of instruction is that when subjects are forced to represent the entities of a task in a verbal form (e.g. when they read exact task instructions concerning those entities) the advantage of expressing the object first is lost. The subjects are able to recruit the verbal representation directly to their task goals. When doing tasks motivated by their own volition, such as in a 'natural' office setting, or when very general task instructions are given; subjects might represent the entities of the task in some non-verbal form which appears to be compatible with an object first syntax.

If this view were accepted, then the reason for the advantage of expressing the object first requires explanation. One possibility, based on a notion from the psychology of language, recruits the concept of topic assignment. In language understanding, it has been proposed (Clark & Clark, 1977) that speakers and listeners agree to a contract where they express 'Given' (or already known) information first followed by 'New' or novel information. It could be conjectured that when performing tasks, users recruit something from the task expression to be the topic of the task (or its superordinate goal), e.g. for the task, 'edit the letter' the user might recruit the letter to be the topic of the task by thinking, 'I'm dealing with the letter'. According to this explanation, the results of the two studies would be expected to differ because the nature of their instructions has an influence on the assignment of a topic. It is reasonable to believe that a general task instruction might be more conducive to the user formulating a topic (as the information must be processed more fully) than a simple and literal instruction where a pattern matching strategy might suffice. Explaining the results in terms of topic relies on the assumption that under the conditions of the observational study objects are more likely to be recruited to the topic than actions. (The concept of a topic will be discussed more fully in Chapter 7).

A further possibility related to the instructions is that in attempting to achieve device-independent instructions for the observational study, a bias favouring objects was present. Usually a stable object descriptor was used (e.g. 'letter'; 'memo') while a number of general purpose action words were used (e.g. 'find' for 'type' in TopView and 'open' in Lisa; 'complete' for 'edit' etc.); while the object descriptors were not the actual names of files (and the word 'file' was not used) it could be proposed that these words have a higher association with the objects in the system than the action words do.
iii) Differences in the functions/semantics of the command words used.
In the observational study, one particular command was used extensively, 'Open'/"Type", the preference towards an object => action syntax might be specifically related to this command. In the experimental study, each of the six commands was employed exactly six times each and there was no 'Look at' command.

Clearly there is scope for further investigation into the conditions which influence the usability of particular command sequence. Given the results of this present experiment, the next step will be to set up alternative experimental conditions in an attempt to re-establish the ease of use of an object-action syntax.

7.2 Modifications to the User Model

It was stated in Chapter 2, that User Models developed from the framework would be expressed in terms of processes and representations concerned with the achievement of task goals. A rudimentary model of this form (comprising a structure and a behavioural model) was proposed in Chapter 5 (see Figures 5.1 & 5.2, Chapter 5). The rudimentary model was based on the premise that object => action syntax is preferred for the computer supported performance of office tasks, as found in the observational study. The findings of the present study will be used to modify this model.

The modified model differs from the rudimentary model in particular respects which reflect the different results found in the two studies. In the present study, it was found that an object => action syntax was not more usable than an action => object syntax. Under the conditions of this study, a device requirement for object => action syntax was not found to enhance representational compatibility between the device and the user.

In the User Structure for this study, it will be assumed that the representation of task goals held by the user is not subject to an ordering process. Thus at the Task Level, the task goals exist in an unordered form such that they exert no bias over syntactic expression at the Communications Level.

The conditions of the study have to be examined to discover why the ordering process is absent. A salient difference between the conditions of this experiment and the observational study lies in the nature of the task instructions. The task in the present study was of a simple and straightforward nature - there was a direct correspondence between the task instructions and the commands required to complete the task. It has been proposed that this form of task instruction obviates the need for the user to assign a topic for the task as a simple pattern
match is adequate. By contrast, the observational study required subjects to comprehend general instructions and then work out which of the computer represented task entities were required to perform the task suggested by the instructions. Often, subjects in the observational study would have to keep in mind what they wanted to accomplish while they searched through items to find an appropriate one. It is expected that some form of topic assignment strategy is implemented under these conditions. Clearly further research into the conditions which influence syntax usage is required. The modification to the user model suggests that some test which explores topic assignment would be informative, as the assumptions outlined here could be evaluated.

If it is assumed that the Task Level ordering process is absent, then no difference in performance of the groups using the two syntax structures would be expected - each condition should be equally compatible with the device. This was true of the two Pick & Point conditions. A User Structure for the Pick & Point conditions is shown in Figure 6.6. (A separate structure is necessary to capture the findings for the command line condition, this structure is described later).

In Figure 6.6, the Task Goals (i.e. the objects and actions) are labelled, x and y. Since the use of objects and actions, with respect to syntax, was equivalent, it is not appropriate to specify which label denotes an object and which denotes an action. Whichever device was used, the user could simply match one of the presented items (either objects or actions) with one item from the Task Goal representation. Also, note that the items in the Task Goal representation match those in the Command representation exactly. The user does not have to translate from a general description (e.g. directory) to a particular one (e.g. A-K telephone directory), as s/he progresses down the levels of description. Due to this exact match between the items described in the task instructions and the items to be specified in the performance of the task, it is likely that conceptual processing was minimal. It would be sufficient for users to recognise a corresponding item, (e.g. 'memo'; 'print' etc.) without having to consider their individual semantics or their meaning in combination, i.e. 'what do you have to do to print a memo' etc.).
Figure 6.6
User Structure for E1 (Pick & Point condition)
The User Structure, shown above, is instantiated in Figure 6.7 to provide a model of a sample of user behaviour. In the example, the given device is one which is non-object-oriented with respect to syntax. In other words, it requires an action => object syntax. However, the user model for the alternative device, (with an object => action syntax), would differ only in the ordering of the commands at the Communications Level and the eventual user actions.

The given device requires input in the form of mouse presses. As in the Rudimentary User Model, user motor programs are abbreviated to MM for a mouse movement and MC for a mouse click.
Figure 6.7
User Model for Pick & Point condition (a->o syntax)
The findings for the command line conditions were different. The object => action syntax group was found to be disadvantaged relative to the action => object syntax group. It is believed that the requirement to type in commands in English, activates interference (or negative transfer) from knowledge of natural English syntax for the imperative. In the object => action condition, the required syntax is directly the opposite of that for the imperative, and so the performance of this group would be expected to be impaired, relative to the performance of the action => object group under this condition. Thus, the subjects performing in the command line conditions are expected to access more detailed knowledge concerning English language, than are the subjects in the Pick & Point condition.

This extra step is shown in the User Structure for the command line condition, see Figure 6.8. This step is expressed in terms of an additional representation, the 'LITERAL STRING', i.e. a precise formulation of the words to be typed. The literal string is expected to be governed by the user's knowledge of English production e.g. spelling rules; syntactic rules etc. As a consequence of the characteristics of a particular device, the user may need to re-express the literal string in order that it is specifies a correct sequence for interaction with the particular device. Thus, for interaction with some devices an additional process will be necessary. Processes and representations which are contingent on the features of a particular device are indicated by dashed lines.
Figure 6.8
User Structure for E1 (Command line conditions)
In Figures 6.9 and 6.10, the structure has been applied to model particular instances of behaviour, as observed under the conditions of the study. In Figure 6.9, the given device is non-object-oriented with respect to syntax, so the ordering constraints determined by the specification of the literal string are consistent with the requirements of the device. In Figure 6.10, the given device is object-oriented with respect to syntax and so the literal string has to be re-ordered to achieve compatibility with device requirements. In both models, the motor programs have been abbreviated such that KP denotes a Keypress.
Figure 6.9
User Model for Command Line condition (action -> object syntax)
7.3 Results Summary

In Section 3 of the present chapter, a need was expressed to determine more precisely the conditions which influence the optimal use of syntax (at the Communications Level). This study has attempted to address that need and has provided a set of empirical findings which concern the use of syntax under varying I/O Level conditions (i.e. the use of different input methods). A useful method of expressing these results would be in the form of a Summary Table, which relies on the levels of description set out in the framework (see Chapter 2). This method of expression is potentially advantageous for two reasons:

i) consideration of the table can be used to guide the direction of future experimental work;
ii) results of any future experimentation can be used to augment or modify the table.

In Table 6.4, the results of the present study are expressed. The table suggests which device features have a consequence for the use of object => action syntax and the nature of that consequence (positive or negative).

<table>
<thead>
<tr>
<th>Task level factor</th>
<th>Communication level factor</th>
<th>I/O level factor</th>
<th>Simple instructions: direct and literal (one object and one action). No topic assignment expected.</th>
</tr>
</thead>
<tbody>
<tr>
<td>topic assignment</td>
<td>o-a</td>
<td>-ve effect</td>
<td>Command style input</td>
</tr>
<tr>
<td></td>
<td>a-o</td>
<td>no effect</td>
<td>Pick &amp; point style input</td>
</tr>
<tr>
<td>Communication</td>
<td></td>
<td>no effect</td>
<td></td>
</tr>
<tr>
<td>level factor</td>
<td></td>
<td>no effect</td>
<td></td>
</tr>
<tr>
<td>syntax</td>
<td></td>
<td>no effect</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>no effect</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.4
Summary table of results
7.4 Results of the Study and the Usability of Object-oriented Interfaces

These findings will now be considered with respect to the question as to whether object-oriented interfaces are easy to use.

A particular dimension, pertinent to the classification of devices, has been isolated, syntax at the Communications Level. Previously (in Chapter 4), it has been suggested that the object-oriented value on this dimension, object => action syntax, aids usability because it renders the device representation of the task more compatible with the representation held by the user. As the advantage of an object => action syntax was previously observed under conditions of Pick & Point input, this study aimed to determine whether the advantage was limited to that style of input conditions. No advantage was found for an object => action syntax under Pick & Point input conditions and in the command line condition, the subjects using an object => action syntax performed worse than those using the action => object syntax. These findings suggest that there is no particular advantage of an object => action syntax in the absence of other factors, i.e. that an object-oriented interface is not easy to use on the basis of its object => action syntax requirement. Inspection of the different conditions of this study and the observational study suggest that factors at the Task Level might interact with the required syntax to have a facilitatory effect on performance.

8. Summary and Conclusions

This chapter reported an experimental study conducted to compare the use of two contrasting syntax sequences and two input styles to perform simple office tasks. The results of a previous study suggested the hypothesis that object => action syntax would be preferred (i.e. would lead to an error-free performance) in all instances. The required syntax and input style were varied (between subjects), but all other factors remained constant (e.g. object:action ratios; actual command names used; medium of object presentation etc.).

Contrary to expectations the object => action, command-line group made significantly more errors than all other groups and significantly more syntax errors than the corresponding action => object group while there was no difference in performance for the two syntax groups of the Pick & Point input condition.

The difference between the two object => action groups has been explored and it has been tentatively suggested that the degree of recourse to English language is responsible. It is assumed that the command line group has to rely on their knowledge of English language to a greater extent than the Pick & Point group and so they experience negative transfer.
The difference between the results of this study and those of the observational study is discussed and three options regarding its cause are considered:
i) the greater degree of control in this study;
ii) the type of instructions given;
iii) the semantics of the actual task entities.
Each of these options constitutes a plausible explanation and they cannot be distinguished on the basis of this study alone, therefore there is a need for further explanation.

The results have been incorporated into a User Model and then expressed in a Summary Table. The expression of the results in this form suggests directions for future experiments.

The User Model includes an important modification in that the representation of task goals held by the user is not expected to undergo any ordering process. The validity of this assumption should be assessed, in other words the conditions for topic assignment should be explored and its effects verified.

The Summary Table shows the effects of I/O Level factors on syntax, an obvious choice for a future experiment is to explore factors at the Task Level. Since topic assignment is a factor which operates at the Task Level, this is a suitable candidate for "filling in" the picture.

In conclusion, this study has shown that the I/O Level factor, input technique, does have an influence on syntax usage. The findings of this study cannot be fully explained and further experimentation is required to a) discover more about conditions which affect syntax usage and b) to examine the conjectural explanations which have been offered in the chapter.
1. Introduction

This chapter reports a study which investigates a conjecture proposed in the preceding chapter. In general terms, the study explores whether a particular task level factor interacts with syntax structures to influence usability.

It has been proposed, in Chapter 6, that in performing tasks users might assign certain task elements to a privileged role, that of being the topic of the task. The topic expresses what the task is about and is its central focus. The notion of topic is explored by considering some of the psycholinguistic literature.

As topic assignment cannot be manipulated easily, with any degree of control, a different, but related, variable is selected for study. This variable, task structure, is expected to provide an indirect index of the effects of topic assignment.

A study, based on the approach described in Chapter 5, is designed. Two groups of subjects are tested on interfaces which differ only in respect of their syntax. The interface and its variations are described.

The design and method of the study are reported. An experimental hypothesis is derived from the conjecture that Task level factors, such as topic or task structure, will influence usability of particular syntax sequences.

The results are analysed and discussed in terms of the experimental hypothesis. Following this discussion, the findings are used to further the analytic activity by enhancing the User Model. The results are also used to expand the Summary Table set up in Chapter 6.

The results of the study suggest that an object=>action syntax is not advantaged inherently, but instead, either an object=>action or an action=>object syntax can be favoured depending on the conditions of task structure.
2. Aims of the Study

The aim of the present study is to extend the investigation of factors which might influence users' performance with alternative syntax structures. It is of particular interest to find out which factors are associated with the enhanced usability of an object => action syntax. In the previous study, in which Task Level factors were held constant, no advantage of an object-> action syntax was found. It was suggested that factors at the Task Level (such as topic) might determine which syntax structure the user finds most compatible with his or her representation of the task.

3. Background

Two previous studies have been performed, one observational (see Chapter 4) and one experimental (see Chapter 6). A discrepancy was found between the results pertaining to syntax usage in the two studies.

In the observational study it was found that subjects using a non-object-oriented interface [IBM TopView] preferred to use an object => action syntax (60.75% instances of syntax usage) when performing office tasks. In the experimental study, which was designed to investigate syntax usage [object => action versus action => object] at the Communications Level under two conditions of command input [Pick & Point style versus Command line], no advantage for an object => action syntax was found. There was no difference between the performances of the two syntax groups in the Pick & Point condition and in the Command Line condition, the performance of the object => action syntax group was worse than that of the action => object group. In order to account for the lack of advantage for the object => action syntax in the experimental study, the conditions for the two experiments were examined in detail.

Several differences were found and, in particular, differences in the task instructions were thought to be important. In the observational study, the subjects were given instructions expressed at a general level i.e. in a manner which did not reflect the actual steps or command words necessary to perform the task using the computer. The subjects were required to work out how to perform the task using the computer. Also, the tasks were fairly complex, often requiring a number of steps. In the experimental study, for the sake of control and to enable the collection of timing data, the tasks were straightforward (each having only one step) and were expressed literally i.e. the actual command names were used. Such a difference is thought to be important as it may have consequences for the way that the user thinks about and represents the task.
It was proposed in Chapter 6, that for complicated office tasks the user formulates a superordinate goal. This goal indicates what the task is about; it is the topic of the task. For example, if a letter is to be edited, copied and printed out for sending, the user might consider the task to be focussed around the letter, with the other aspects of the task forming the background. By contrast, if s/he needs to file away a memo, a report, a graph and a table, the user might think the task to be one of 'filing'. Topicalisation is expected to have consequences for syntactic expression. In particular, it is thought that users will tend to express the topic first. For example, if the user designates a letter as the topic of a task, then s/he will tend towards expressing 'letter' as the first element of a command sequence, as in 'letter edit'; 'letter print'; 'letter copy' etc.

This strategy of expressing the topic of a task first has similarities with the Given-New strategy found in the psycholinguistics literature (Clark & Clark 1977; Haviland & Clark 1974) where a speaker expresses the 'given' or assumed known information first, followed by new or novel information. It is proposed that there is an implicit contract between the speaker and listener that information will be organized in this way. Note that the Given-New distinction usually applies to referents as perceived by the listener; Halliday (1967) has used the terms Theme and Rheme to apply to a similar construct from the speaker's point of view. Halliday asserts that the Theme (what the utterance is about) always precedes the Rheme. However, there is considerable debate in the psycholinguistic literature concerning the exact nature of such constructs.

Chafe (1976), attempts to elucidate this area. He takes six noun statuses which are commonly used by linguists and describes them so as to pinpoint their meanings. Of the six statuses, three are of interest: 'Giveness', 'Subject' and 'Topic'. Chafe comments that an item will have the status 'given' if the speaker assesses that it already exists in the listener's mind. Clearly this status has more to do with the speaker's knowledge and on-going assessment of the listener and their interaction than with the meaning of the utterance for him/herself. However it would seem that it has an influence on syntactic choices. The status 'subject' appears to have a number of different meanings for different people. Chafe considers the status 'subject' to be assigned to items about which new knowledge is being added. Hornby (1971, 1972) in his work on the psychological subject has said that, "the part of the sentence which constitutes what the speaker is talking about is being called the TOPIC of the sentence . . . the rest of the sentence, the COMMENT, provides new information about the topic" ('topic' here is equated with 'subject') Chafe believes that Hornby has provided a temptation to confuse 'subject' with 'given'; when there is no reason for the subject to entail given information. Chafe suggests that the notion of 'topic' is not straightforward, he believes it to differ in different languages with some languages being more explicit about topic assignment and notification than others. The notion of topic is difficult to explain with
reference to English because this is not a topic-prominent language. In such languages, e.g. Chinese, it is believed that the topic is stated initially and this limits the applicability of the subsequent phrase. Chafe states that, in English, topic can be equated with the sentence-initial position of certain contrastive items. For example, in "The play, that John saw yesterday", Chafe asserts that 'the play' is a focus of contrast, i.e. there is an implicit list of items against which the play has been judged to be the thing which John saw - it was a play as opposed to a football match, a television programme, a lecture etc.

On comparing the notion of 'topic' (as described above) with the statuses described by Chafe, it would seem that it corresponds most nearly with 'subject'. Chafe states that, in English, a subject is 'what the sentence is about'. However, while taking note of these distinctions, the term 'topic' will be retained for ease of reference. The term is used here in the context of the focus of a task and so should not be confused with constructs in psycholinguistics.

The psycholinguistic literature suggests that assignment of status (such as subject) is likely to have consequences for syntactic expression. In particular, it is found that the subject/topic often assumes an early position in the sentence, as if establishing a focus. Bock's work on syntax proposes that the referential arena, where non-verbal thought processes are entertained, has an influence on syntactic expression and this is reflected in her model of syntactic processing (Bock 1982). If topic formation is considered to be a non-verbal thought process which determines the ordering of goal representations, then it is reasonable to expect that it would influence the way in which a user would tend to behave with respect to syntax when interacting with a device. For example, having determined that a task concerns a memo, i.e. establishing a topic, a user might be predisposed to specify the object (the memo) first in subsequent expressions of task elements.

However, at this stage it is unclear why objects should have a greater tendency than actions to be assigned to this topic role, which has been proposed to explain the advantage of an object => action syntax.

In a different domain, that of interface development, Tullis (1985) made an interesting discovery. He was attempting to determine an appropriate organization for a range of functions which were to be incorporated in a menu style interface. Towards this end, he conducted a card sorting task. It was found that subjects grouped the functions by the object which they involved rather than the action. Tullis assumed that in using the functions the users attentions were focussed on the objects rather than the actions.

It seems that topic assignment might exert some influence over the usage of syntax, though
the mechanism responsible and the nature of the influence are unclear. Therefore, an empirical investigation focussing on the effects of topic assignment on syntax, is justified.

However, the empirical study of topic effects is beset with problems. It is difficult, if not impossible, to have an objective measure of whether an appropriate topic has been assigned by a subject. To enquire via language may result in a response reflecting task instructions and the enquiry might influence the subject's subsequent behaviour.

Due to such problems, a decision was made to obtain an indirect index of the effect of topic assignment by using as a variable, a measurable task factor, task structure. Task structure refers to the order of execution of the sub-tasks of a task e.g. if a task requires four operations to be performed on four objects, then there are at least two methods of execution: each object can have all four operations performed on it before progressing to the next object:

OBJECT-ORIENTED STYLE (O1-A1; O1-A2; O1-A3; O1-A4; O2-A1; O2-A2 ... etc.)

or a single operation can be performed on all four objects before progressing to the next operation:

ACTION-ORIENTED STYLE (A1-O1; A1-O2; A1-O3; A1-O4; A2-O1; A2-O2 ... etc.).

Jorgensen and Barnard (1986) report a study in which the use of the two structures is compared for messaging tasks. They found that the object-oriented structure was easier to learn in terms of time, errors and requests for help. Their study did not address the effects of task structure on user performance with alternative command sequences.

Here, the task structure will be varied to induce assignment of object topics and action topics. It is assumed that if a user subject is given a task requiring a number of operations to be performed on the same object (e.g. edit letter, file letter, print letter, copy letter etc.) then the user's model of the task will approximate the model for a task where an object topic is assigned naturally (and similarly for an action-oriented structure). Note that in the observational study, the tasks tended towards an object-oriented task structure. The aim of the study is to investigate whether this variation in task structure has an effect on the use of command syntax. It is expected that performance will be facilitated when the task structure and the required syntax are consistent e.g. an object-oriented task structure, where a number of actions are performed on a common object (copy memo; edit memo; send memo; file memo), will be executed more easily if the required syntax is object => action as the repeated item (topic) is placed first in the command sequence. Additionally, it is of interest to find out whether subjects tend towards consistency between the task structure and syntax under free
syntax conditions (where they can use the syntax of their choice).

The study is reported in the following sections.

4. Details of the Simulated Interface

The method for experimentation given in Chapter 5 was followed, i.e. a simulated interface was employed as the test interface. As before, the simulation, written in HyperTalk, was run on an Apple Macintosh II. All versions of the system had the following general features.

The first screen was an 'option' screen which presented the available objects and actions to the subjects. After each task (successful or unsuccessful), the subjects returned to this screen. The option screen had two sections, one above the other, one section presented all the objects and the other all the actions. A command sequence was specified by choosing one object and one action, in the correct order. Subjects could choose an item by clicking the mouse button while the cursor was positioned over it.

If the incorrect order was used (in the fixed order systems), a message appeared at the bottom of the screen. The message read, 'Incorrect command sequence - please click to continue' and the subjects were presented with a button labelled 'Continue'. Depressing the 'Continue' button enabled the system to execute the command as if it had been entered in the correct order. The subjects did not have to go back and specify the command again. This arrangement was used deliberately:

i) to provide the subjects with feedback when they had made a mistake (to avoid encouraging a careless attitude to syntax usage);

ii) to avoid discouraging subjects from acting naturally (even if it involves making mistakes).

In an experimental situation, where subject motivation and concentration are high, subjects avoid actions which result in highly negative consequences. Often subjects are willing to adopt an avoidance strategy even if it results in them taking longer to complete the task and expending greater effort. So the behaviour which is recorded is more an index of the subjects' desire to avoid negative consequences, than it is a demonstration of the subjects' natural response to task demands.

Each system had the following 10 objects:

<table>
<thead>
<tr>
<th>Letter</th>
<th>Table</th>
<th>Chart</th>
<th>Record-card</th>
<th>(d) Diary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memo</td>
<td>Report</td>
<td>Graph</td>
<td>Message</td>
<td>(d) Directory</td>
</tr>
</tbody>
</table>
and 10 actions:

<table>
<thead>
<tr>
<th>Print</th>
<th>Copy</th>
<th>Send</th>
<th>Compact</th>
<th>(d) Archive</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Rename</td>
<td>Edit</td>
<td>Display</td>
<td>(d) Retrieve</td>
</tr>
</tbody>
</table>

(d) = a distractor item which was not used in any of the tasks.

Distractor items were included to prevent subjects from anticipating that each of the on-screen items would be used once.

In the simulation, there was only one instance of each object i.e. a single letter, memo etc. While the deviation from the real-world case is recognised this arrangement was chosen for two reasons:

i) it simplifies the syntax;

PRINT LETTER or LETTER PRINT

there is no need to specify a particular instance as in:

PRINT LETTER [LETTER X] or LETTER [LETTER X] PRINT etc.

The alternative solution of representing a number of particular objects separately would preserve the simple syntax but would disrupt the balance between the number of objects represented and the number of actions represented.

ii) to obviate the need for navigation. Commonly systems which represent a number of objects arrange them in some hierarchical fashion (directories; folders etc.); finding one's way around such systems is often difficult, particularly for the novice (as found in the TopView group in the observational study). By having only one object of each type, possible confounding effects of navigation are avoided.

After the test session, this reasoning was explained to the subjects.

An illustration of an option screen (with the actions displayed in the top panel) is shown in Figure 7.1.
There were a number of requirements that the simulated interface had to fulfil. It was necessary that a number of different versions of the interface could be generated that differed only in specified ways. There had to be three versions of the interface which differed in their syntax to yield the two groups object => action syntax and action => object syntax and a further version for the second test session which allowed either syntax sequence to be used, the 'free' syntax interface.

Also, different versions had to be generated to provide balance of presentation between the groups. The factors to be balanced were:

i) arrangement of objects and actions on the screen (objects in top section; actions in top section)

ii) arrangement of particular command words (at least two different arrangements were needed)

All versions of the interface represented the functionality in the same way, i.e. they were identical apart from the initial 'option' screen. Also, all versions had a data capture facility which recorded the commands selected and the time taken to select the first command.
5. Method

5.1 Design

The experiment was conducted in two sessions. In the first session, subjects used an interface which had a fixed syntax and in the second, they used an interface with free syntax (i.e. the subject could choose commands in any order which was convenient). This scheme was chosen because in previous studies (Barnard et al 1982), it has been shown that free syntax systems require more learning effort. Here, subjects would already be accustomed to the system before they encountered the free syntax condition.

In the first test session there were two independent variables:

Required syntax (object => action; action => object)
Task structure (object-type; action-type)

For each subject, the required syntax was constant for the entire test session. Half the subjects had an object => action syntax and half had an action => object one. Subjects were informed of their particular syntax in the instructions for the test session. The task structure was varied so that each subject received object-type tasks on half of the trials and action-type tasks for the other half. This variation meant that on half the trials the task structure and the syntax were consistent. If a subject using an object => action syntax interface received an object type task, such as:

MEMO
You want to re-use this memo, so make a copy of the original, called "Memo/Storage".

the task type and the required syntax would be consistent - the subject would specify the command sequence, 'memo-copy'.
If the same subject had received an action type task, such as:

PRINT
You need two draft quality print-outs of the memo to stick up on noticeboards around the department.

the two variables (task structure type and required syntax) would be inconsistent - the subject would have to specify the command sequence, 'memo-print'.

There were two phases of the experiment with sixteen subjects being used in each phase. The
phases differed in their order of presentation of the two task types. In phase one, subjects 1-16 received a block of six trials of one type followed by a block of six trials of the other type. In phase two, subjects 17-32 received a mixed set of twelve trials following a pattern of alternation between the two types (e.g. one object-type; two action-types; two object-types; one action-type etc.). It was believed that subjects in the first phase had an opportunity to adopt a proceduralised response strategy and after a few trials, errors due to an inconsistency between task structure and syntax would be unlikely. There would be only one opportunity to observe what happens when the procedure became redundant i.e. when the task structure was changed (Task 7).

There were three dependent variables:
Errors
Difficulties or sub-optimal responses
Time taken to select the first element of a command

There were a number of balancing factors between groups. These were:

i) the position of the action set and the object set on the screen - for half the subjects the objects appeared in the top section and for the other half the actions appeared in the top section.

ii) the position of each particular command word within the set - there were two arrangements of the command word sets so that words that appeared in the centre position on one set appeared at the periphery in the other set.

iii) the sequence of tasks used - for all phases there were two sequences of the tasks.

Each task was composed of four object-action pairings from the set of sixty-four possible pairings (8 objects x 8 actions); with no pairing being repeated within the set of tasks. Each subject was given twelve tasks, so a total of forty-eight object-action pairings were performed in the first test session. Details of the tasks are given below.

In this first session, it was of interest whether consistency between the task structure and syntax facilitated performance. If this was the case, one would expect greater syntax error and difficulty scores on inconsistent trials, as well as greater response latencies in selecting the first command because subjects would have to apply additional processes to translate their representation into an appropriate form.

In the second test session, there was one independent variable:

Task structure (object-type; action-type)
Here, the subjects experienced a single structure for the entire session; this was object-type for half the subjects and action-type for the other half. Each subject had eight tasks, composed of four sub-tasks (32 pairings).

The subjects were told that they were to use a slightly different computer system. The content of the actual items differed in the second test session - e.g. the letter was concerning a different subject and it was addressed to someone different. Subjects were instructed to choose the commands in any order they liked and their preference was scored as a dependent variable. The extent to which the syntax chosen was consistent with the task structure was measured.

Expressed generally, the experimental hypothesis asserts that:

\[
\text{a particular syntax sequence should be easier to use when it is compatible with the task structure, i.e. when the first element of the sequence expresses the repeated item of the task structure.}
\]

5.2 Subjects

Subjects’ participation in this study was voluntary and each subject was paid £2. There were 32 subjects (17 female and 15 male). The mean age of the subjects was 23.6 years (sd = 7) with a range of 17-47 years. The majority of the subjects were students and research staff from University College London. All but 6 of the subjects spoke English as their first language. Only one subject had no previous computer experience. The other 31 subjects had used computers for wordprocessing, data analysis, running experiments and playing games. Half the subjects had used a mouse before.

5.3 Domain

The domain of common office tasks was chosen, as for previous experimentation. It includes the set of common tasks undertaken by professional workers regardless of their particular area of expertise e.g. filing letters, writing memos, producing graphs, keeping diaries etc. The domain remained constant for all groups.

5.4 Tasks

As described above, the set of tasks was devised from combinations of the 8 objects and the 8 actions. There were two types of task: object-type tasks and action-type tasks. The
object-type tasks were focussed around one object which was to have four operations performed on it whereas the action-type tasks focussed on a particular operation which was to be performed on four objects.

The manner of presentation of the task instructions was the same for all tasks. The subject was shown a card by the experimenter. The card contained 8 options (objects or actions) with one option highlighted. The subject was asked to keep the highlighted option in mind (as this was what the task was about). Then the subject could look at the details of the task in a folder - the instructions for each task were typed on a separate page. By combining the highlighted item and the instructions the subject could work out what was required.

Twelve tasks were used in the first test session and eight in the second session. More tasks were included in the first test session to compensate for the subjects' unfamiliarity with the device, in general.

See Appendix H for a complete description of the task set. Sample tasks appear below:

Object-type task  (the highlighted option, 'Memo', is the 'topic' of the task)

Directory
Memo
Table
Letter
Record-card
Graph
Report
Message

You want to re-use this memo, so make a copy of the original, called 'Memo/Storage'
Now you've had an enquiry about the market research material from DM Cherry, edit the name of the recipient and the date so it is ready to send to him - then send it to him
You like to group together all material of an organisational nature under the heading, 'Admin', put the memo under this heading
So the user could work out that for an object=>action syntax system the four command sequences were:
MEMO-COPY; MEMO-EDIT; MEMO-SEND; MEMO-FILE

Action-type task (the highlighted option, 'Print', is the 'topic' of the task)

---

You need two draft quality print-outs of the memo to stick up on noticeboards around the department.
The report is going to be bound with the quarterly reports, so you should make a best-quality print-out which can be dispatched to Publications.
The graph is also going to be included in a publication so make three best-quality, enlarged print-outs.
Make a draft quality print of the message to show to the Production Team.

Here the user could work out the sequences:
MEMO-PRINT; REPORT-PRINT; GRAPH-PRINT; MESSAGE-PRINT

5.5 Training

Each subject was given some basic training. The training materials were written in HyperCard and were intended to provide the subjects with the necessary interaction skills to perform the tasks e.g. using the mouse for selection; typing into response boxes; basic editing.
skills etc. The training was unrelated to the tasks themselves so subjects were expected to apply and combine the skills they had learned when confronted with the experimental tasks.

After the training, the subjects watched an on-screen demonstration (again written in HyperCard). The demonstration showed an interaction with an interface resembling the one to be used in the test session. The demonstration system used numbers {1, 2, 3, etc.} and letters {a, b, c, etc.} for the commands instead of object and action words. The interaction showed the selection of a command sequence, demonstrating that two elements must be selected; one from each section of the screen and they must be selected in the correct order. An error situation was shown, so that the subjects would be aware of the error message and the appropriate recovery. It was pointed out to the subjects that they should not pay attention to the details of the demonstration e.g. the syntax that was used, but that they should concentrate on the principles of the interaction, that two items, one from each half of the screen, should be selected as these would make up a command sequence.

5.6 Questionnaire

After both test sessions had been completed, subjects were asked to fill in a questionnaire. The questions were arranged in two parts. The first part asked about personal details (age; sex; occupation etc.) and previous computer experience, while the second part addressed the subjects' reactions to the test sessions. The questions in the second part usually required the subject to select an appropriate response from those given (often 'Yes' or 'No').

5.7 Experimental Session

The experimental session for each subject, irrespective of their group, followed this scheme:

1 General Instructions
2 Training & Demonstration
3 Test Session I Instructions and System Description
4 Test Session I
5 Test Session II Instructions
6 Test Session II
7 Questionnaire

The overall session lasted for approximately 1 hour.
See Appendix I for details of the instructions and questionnaire.
6. Results

The dependent variables, errors, difficulties and time, were recorded. The results from the two phases of the experiment were analysed separately (i.e. Subjects 1-16 and Subjects 17-32). The results were analysed in a number of ways. For the first test session, where the subjects had a fixed command syntax (object-action or action-object), three measures were analysed: syntax errors; difficulties or sub-optimal responses and time to select the first command. In the second test session, a single measure was analysed: preferred command syntax. The results of the analyses are described below.

6.1 Test Session 1

6.1.1 Syntax errors

The number of times that a subject selected the correct command items in the wrong order was recorded. It was of particular interest to find out whether the errors occurred on trials when the task structure and syntax were consistent or inconsistent. The hypothesis predicts that more errors will occur on inconsistent trials.

Mean scores for each group appear in the following tables. An ANOVA has been performed on the data with a between subjects factor, syntax, and a within subjects factor, task structure.

Also a Mann-Whitney U test has been used to compare the scores that were obtained on consistent trials with those contained on inconsistent trials - this combines the two variables.
Subjects 1-16

<table>
<thead>
<tr>
<th>Task Structure</th>
<th>Object type</th>
<th>Action type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td></td>
<td></td>
</tr>
<tr>
<td>object =&gt; action</td>
<td>8 (1)</td>
<td>2 (0.25)</td>
</tr>
<tr>
<td>action =&gt; object</td>
<td>1 (0.125)</td>
<td>0 (0)</td>
</tr>
</tbody>
</table>

Table 7.1
Number of syntax errors (S1-16)

For this group the overall number of syntax errors was not very high. It is important to point out that of the total of 11 errors, 9 occurred on the first trial. Neither the main effects nor the interaction were significant, (p > 0.05).

[Syntax F (1,14) = 4.3; Task structure F(1,14) = 3.45; Interaction F (1, 14) = 1.829].

Mann-Whitney U test comparing the scores for consistent trials and inconsistent trials was not significant, (p > 0.05).

Subjects 17-32

<table>
<thead>
<tr>
<th>Task Structure</th>
<th>Object type</th>
<th>Action type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td></td>
<td></td>
</tr>
<tr>
<td>object =&gt; action</td>
<td>1 (0.875)</td>
<td>5 (0.625)</td>
</tr>
<tr>
<td>action =&gt; object</td>
<td>4 (0.5)</td>
<td>0 (0)</td>
</tr>
</tbody>
</table>

Table 7.2
Number of syntax errors (S17-32)
Again the overall number of syntax errors was not high. However, they were more evenly distributed between tasks. Neither the main effects nor the interaction was significant, (p>0.05), [Syntax $F(1, 14) = 0.823$; Task structure $F(1, 14) = 2.334$; Interaction $F(1, 14) = 0.259$].

Mann Whitney U test comparing the scores for consistent trials and inconsistent trials was not significant, (p >0.05).

6.1.2 Difficulties or Sub-optimal Responses

Sometimes a sub-optimal sequencing response was found to occur during the selection of the command sequence. This response was recorded as a dependent variable.

The command sequences always consisted of two elements; an action and an object. During the first test session of the experiment, each subject was required to specify the command sequence in a particular order.

If the required order was action => object, then the correct sequence for filing a letter would be:

FILE LETTER

and the error case would be:

LETTER FILE

A sub-optimal sequencing response would be scored if the subject distinctly moved the cursor to rest on LETTER, paused (as if about to select it and make an error) and then (as if s/he remembered the system requirements) moved directly to FILE and selected it and then moved back to LETTER and selected it.

The necessary conditions for a sub-optimal sequencing response are therefore:

- a distinct movement of the cursor to the second element of the command sequence.
- a pause with the cursor resting on the second element of the command sequence.
- a rapid movement directly to the (correct) first element of the command sequence, followed by selection of the first element.
- a movement back to the second element of the command sequence, followed by its selection.

In general, sub-optimal sequencing responses were noted during the actual test session. They
were then checked and verified from the video recording.
It was of interest whether this partial error case was related to the type of trial (i.e.
consistent or inconsistent). It was expected under the experimental hypothesis, that a higher
number of these responses would occur on inconsistent trials.

The following tables, 7.3 and 7.4, show the number of these responses scored for each group.

Subjects 1-16

<table>
<thead>
<tr>
<th>Task Structure Syntax</th>
<th>Object type</th>
<th>Action type</th>
</tr>
</thead>
<tbody>
<tr>
<td>object =&gt; action</td>
<td>1 (0.125)</td>
<td>3 (0.375)</td>
</tr>
<tr>
<td>action =&gt; object</td>
<td>2 (0.25)</td>
<td>0 (0)</td>
</tr>
</tbody>
</table>

Table 7.3
Number of sub-optimal responses (S1-16)

No significant effects were found of syntax, task structure or an interaction, (p >0.05).
[Syntax F (1, 14) = 0.466; Task structure F (1, 14) = 0; Interaction F (1, 14) = 2.793]
Mann Whitney U test comparing the scores for consistent trials and inconsistent trials was
not significant, (p >0.05).
Table 7.4
Number of sub-optimal responses (S17-32)

Neither of the main effects, syntax or task structure, was found to be significant, (p >0.05).
However, the interaction (syntax x task structure) was significant, p <0.05.

Mann Whitney U test comparing the sub-optimal sequencing response scores for consistent and
inconsistent trials was not significant, (p >0.05).

6.1.3 Combining Error and Sub-optimal Sequencing Response Scores

It was of interest whether there was an effect of task structure on syntax when the effect is
evidenced by either errors or partial errors. Here, the data for syntax errors and partial
errors have been combined. Clearly the two error cases are mutually exclusive; either the
error is a syntax error or an almost-syntax error.
Subjects 1-16

<table>
<thead>
<tr>
<th>Task Structure</th>
<th>Object type</th>
<th>Action type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>object =&gt; action</td>
<td>9 (1.125)</td>
</tr>
<tr>
<td></td>
<td>action =&gt; object</td>
<td>3 (0.375)</td>
</tr>
</tbody>
</table>

Table 7.5
Combined errors and sub-optimal responses (S1-16)

Neither of the main effects was significant, (p>0.05), [syntax F (1,14) = 3.87; task structure F (1,14) = 4.349]. The interaction was also not significant, (p>0.05), [F (1,14) <1].

A Mann Whitney U test comparing the combined errors and sub-optimal sequencing response scores for consistent and inconsistent trials was not significant, (p >0.05).

Subjects 17-32

<table>
<thead>
<tr>
<th>Task Structure</th>
<th>Object type</th>
<th>Action type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>object =&gt; action</td>
<td>8 (1)</td>
</tr>
<tr>
<td></td>
<td>action =&gt; object</td>
<td>8 (1)</td>
</tr>
</tbody>
</table>

Table 7.6
Combined errors and sub-optimal responses (S17-32)

The main effects were both not significant, (p >0.05), [syntax F (1,14) = 1.22; task structure F (1,14) = 0.1106]. However, the interaction (syntax x task structure) was significant at p <0.05 [Interaction F (1,14) = 7.08].
A Mann Whitney U test comparing the combined scores for consistent and inconsistent trials showed a significant difference between scores for the two trial types at \( p < 0.01 \) [\( U = 66 \)].

The interaction is depicted in the following graph of the mean scores, Figure 7.2. The asymmetrical nature of the graph will be considered in a later section.

![Graph of mean error and sub-optimal response scores (S17-32)](image)

**Figure 7.2**
Graph of mean error and sub-optimal response scores (S17-32)

### 6.1.4 Timing

The time from the presentation of the option screen to the selection of the first command was recorded (in 1/60 secs) for every sub-task. It was expected that there would be a greater response latency on inconsistent trials. A computer-aided (SPSS PC+) analysis of variance was performed on the two sets of time data.
Subjects 1-16

<table>
<thead>
<tr>
<th>Task Structure</th>
<th>Object type</th>
<th>Action type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td></td>
<td></td>
</tr>
<tr>
<td>object =&gt; action</td>
<td>9.4896</td>
<td>9.5705</td>
</tr>
<tr>
<td>action =&gt; object</td>
<td>9.3332</td>
<td>10.0746</td>
</tr>
</tbody>
</table>

Table 7.7
Mean time to select first command (S1-16)

The analysis of variance showed that there were no significant main effects [syntax F(1,805) = 0.088; task structure F(1,805) = 0.48] and no significant interaction, syntax x task structure [interaction F(1,805) = 0.58], (p >0.05).

The time to select commands did not differ with task structure for this group of subjects. No effect of task structure can be assumed.

Subjects 17-32

<table>
<thead>
<tr>
<th>Task Structure</th>
<th>Object type</th>
<th>Action type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td></td>
<td></td>
</tr>
<tr>
<td>object =&gt; action</td>
<td>10.4689</td>
<td>13.5043</td>
</tr>
<tr>
<td>action =&gt; object</td>
<td>11.8949</td>
<td>11.4547</td>
</tr>
</tbody>
</table>

Table 7.8
Mean time to select first command (S17-32)

The analysis of variance showed that the main effects were not significant, (p>0.05), [syntax F(1,804) = 3.626; task structure F(1,804) = 0.205]. However, the interaction, syntax x task structure, was significant at p <0.011 [interaction F (1,804)= 6.544].
This result suggests that task structure interacts with syntax to influence performance. A task structure which is inconsistent with the required syntax results in longer times to select the first command of the sequence. This effect occurs whether the required syntax is object-action or action-object. It is interesting to note that the effect is asymmetrical as there is less of a detriment in performance for the object-action syntax group on inconsistent trials. This finding will be considered later.

The means are depicted in the following graph, Figure 7.3:

The timing data described above were derived by scoring every instance of command usage for each sub-task, whether or not it was correct. An additional analysis of variance was performed on a set of 'corrected' time data, where only one command per sub-task was included. Here, only the last attempt for any sub-task was recorded and non attempts were scored as missing values. The pattern of results of this ANOVA were similar to those of the previous test with the main effects being non-significant (syntax F (1,755) = 2.579; task structure F (1,755) = 0.327) and the interaction, syntax x task structure, being significant at p<0.18, (interaction F (1,755) = 5.642).
6.2 Test Session 2

In the second test session the syntax used by the subjects on each trial was recorded. It was of interest whether subjects varied their syntax usage in accordance with the task structure presented rather than continuing to use the syntax order required in the first test session.

There were essentially two groups in the second session:

i) a group for which the task structure was consistent with their previous syntax experience (e.g. subjects who had used an action => object syntax in the first session, and who now were given an action type task structure). The 'consistent' group.

ii) a group for which the task structure was inconsistent with their previous syntax experience (e.g. subjects who had used an object => action syntax in the first session, and who now were given an action type task structure). The 'inconsistent' group.

If an effect of task structure was present, it would be expected that subjects of the 'inconsistent' group would modify their syntax usage to accommodate the task structure, whereas those in the 'consistent' group would retain the same syntax.

For each subject, the percentage of instances in which they selected the same syntax as they had used previously was scored. It would be expected that this percentage would be a high value if the task structure and the previous experience were consistent and low value if the reverse were true.

Inspection of the data revealed that there was a trend in the expected direction. However, a Mann-Whitney U-test, comparing the scores for the 'consistent' group with those for the 'inconsistent' group showed the difference between the groups was not significant, \((p>0.05)\), \([U = 88.5]\).

6.3 Questionnaire Responses

The questionnaire was arranged in two parts. The first part concerned subjects' personal details and previous computing experience while the second part concerned their reactions to the test session.

Results of the responses to the second part of the questionnaire are summarised below.

i) General questions
- 93.75% of subjects thought that the interfaces were easy to use.
- All subjects thought that the second test session (free syntax) was easier than the first, or that there was no difference between sessions.
- 97% of subjects found the command words (objects and actions) easy to understand.
- Only 15.6% of subjects reported difficulty in using the mouse, although only half of the subjects had encountered a mouse before.
- 87.5% of subjects thought that the interface which they had used was either 'Very Useful' or 'Quite Useful'.

These responses show that the majority of subjects reacted positively to the interfaces which they had used, irrespective of the particular variant in use. It should be mentioned that many of the subjects believed that the interface was an extant application package and were surprised when they were told it was a simulation.

ii) Questions about performance
- 34% of subjects thought that they had made syntax errors during the test session. This figure is representative of the actual number of errors that were made.
- 12% of subjects found it difficult to use the syntax which was required in the first test session, (9% were in an object => action syntax group and 3% were in an action => object group).
- 25% of subjects did not like the syntax which they were required to use, (18.75% were in an object => action syntax group and 6.25% were in an action => object group).
- When asked to write down all the command words that they could remember (maximum possible score = 20), the mean number of correct responses was 13.4 (SD=3.12), with 66% of subjects being able to remember more actions than objects.
- When asked which syntax they would prefer to use, 31.25% selected object-> action; 37.5% selected action => object and 31.25% selected a mixed syntax.

These responses show no convincing evidence that subjects thought one particular syntax was easier to use or preferable to another.

7. Discussion

The analysis of the results has shown that there was no significant effect of task structure on syntax for subjects 1-16 on measures of errors; sub-optimal responses; combined errors and sub-optimal responses or time to select commands. For subjects 17-32, there was no significant effect of task structure on measures of errors or sub-optimal responses. However, when error and sub-optimal response scores were combined there was a significant interaction between task structure and syntax at the p<0.05 level (interaction F (1, 14) = 7.08). The interaction,
task structure x syntax, was also significant (at p<0.011) when the times to select the first command were analysed (interaction F (1,804) = 6.544). It seems that when task structure and syntax are inconsistent, subjects make more syntax errors and partial syntax errors and take longer to select the first command than when task structure and syntax are consistent.

Analysis of the results for the second test session, showed that subjects given a task structure inconsistent with their previous syntax experience, would tend to modify their syntax usage to achieve consistency. However, this finding was not statistically significant, (p >0.05) [Mann-Whitney U-test, U=88.5].

7.1 Implications of Results

7.1.1 Difference between Subjects 1-16 and Subjects 17-32

The results reported showed that there was a difference in performance between the subjects having a block presentation of task structures (subjects 1-16) and the subjects having a varied presentation (subjects 17-32).

For all subjects, the incidence of errors and difficulties was low. It can be assumed that this is due to the low task demands of the experimental set-up. In a laboratory setting, one can expect that subjects will be unusually highly motivated to perform well (providing that the task is not unduly repetitive or unpleasant). Laboratory subjects are aware that their performance is under scrutiny. Additionally, they have no external influences which might adversely affect their performance. This situation contrasts with office workers in the real-world, whose performance (at an operational level) is not monitored and who are subject to interruptions and conflicting task demands. Despite this low incidence of scorings on the dependent variables, some significant effects were found for subjects 17-32 (significant interactions occurred for combined errors & sub-optimal responses and time to select the first command), while no significant effects were found for Subjects 1-16.

Subjects 1-16 were exposed to one task structure for half the trials (1-6) and the other task structure for the remainder of the trials (7-12). Typically, these subjects made errors on the first trial, but then appeared to grasp a strategy for response to which they adhered. It appeared not to make a difference whether this strategy dealt with consistency or inconsistency between the task structure and syntax. Possibly the lack of a difference in the performance of subjects who had consistent trials compared with subjects who had inconsistent trials, is a result of the generally low task demands.
For this group, it would be expected that any effects of task structure would be evident at the switch point between the two structures i.e. on Task 7. Under the experimental hypothesis, it would be expected that syntax errors would occur if the task structure became inconsistent, when it had previously been consistent. This effect was observed for one subject, Subject 12. No other subject made a syntax error, but for a number of subjects, a marked hesitancy was observed on Task 7 and some subjects remarked on the difference. However, once the changeover had taken place subjects did not seem to experience problems in reversing their strategy.

It could be argued that Subjects 1-16 experienced even lower task demands than Subjects 17-32, as their conditions were more predictable. Under this explanation, it could be proposed that the influence of task structure on syntax becomes more apparent as task demands increase.

7.1.2 Asymmetry within the Subjects 17-32 Group

For Subjects 17-32, significant interactions were found between task structure and syntax on measures of time to select the first command and combined error and sub-optimal response scores. These findings suggest that task structure has an effect on subject performance with particular syntax structures. However, in both cases the effect was asymmetrical (see Figures 7.2 and 7.3 above).

On the time measure, the object-action group were found to be quicker (though not significantly so) than the action-object group on both types of trial. Inconsistent trials had less of a detrimental effect on performance for the object-action group than for the action-object group.

On the combined error and sub-optimal response measure, the object-action group were found to have higher scores overall than the action-object group (no significant difference) and although their scores fell on consistent trials, they did not improve to the same level as for the action-object group.

From this study alone, it is difficult to suggest why the effect was found. It is particularly noticeable that the asymmetry did not follow the same pattern on the two measures. The object=> action group exhibited a favourable performance with respect to time, while the action => object group showed the better performance in terms of errors and sub-optimal responses. However, both these effects are small as the difference between groups failed to reach statistical significance on either measure.
7.2 Modification to the User Model

The User Model which was expressed in Chapter 6 will be developed to include the findings of this study. The most prominent alteration is the incorporation of an ordering process at the Task Level which is governed by topic assignment/task structure. This process operates on the user's representation of task goals. It orders the task goals such that the topic (or repeated task element) is expressed first. At the Communications Level, the user must express the task goals in a form which is compatible with the requirements of the device. The translation entails a consideration of the actual command words present and any syntax constraints imposed by the device. At this point there may be a conflict with the user's representation, if the topic cannot occupy the first position in the command syntax. Under such conditions a detriment in performance is expected (either observed as errors or longer response latencies).

The modified User Structure, shown in Figure 7.4, specifies the user's processes and representations in a general form. In the diagram the labels 'x' and 'y' are used to denote either objects or actions, because in this instance they are interchangeable - the ordering with respect to task structure can favour either an object or an action. As in the structure of Chapter 6, the notation x' is used to show that a common referent can be expressed in difference forms (e.g. x and x').

It should be noted that the general structure includes two alternative routes through the task. The choice between the two routes is contingent on the features of the device. As explained above, if the syntax required by the device is not consistent with that of the user's representation of task goals, an additional re-ordering process is necessary.
Figures 7.5 and 7.6 apply the user structure it model instances of user behaviour under different conditions of the study. Figure 7.5 shows behaviour on a consistent trial where the device is object-oriented with respect to syntax (object => action syntax) and the repeated item expressed in the task is an object (memo).

By contrast, Figure 7.6 depicts behaviour on an inconsistent trial, where the device is non-object-oriented with respect to syntax (action => object) but the repeated item is an object (memo). Here, the user must apply an additional process to re-order the command sequence to satisfy device constraints. Performance under this condition might be expected to be worse than that for the consistent condition as:

a) the user may omit the extra process and so commit an error;
b) the user will take longer to execute the task because there are more steps to perform.

As before in the user models, the motor programmes for user actions have been abbreviated with MM indicating a mouse movement and MC indicating a mouse click.
Figure 7.5
Example of user model for Consistent trial
Figure 7.6
Example of user model for an inconsistent trial
7.3 Results Summary

The Summary Table expressed in Chapter 6, can now be expanded to include the effects of the Task Level factor, task structure. First, the results of the two experimental studies and the observational study will be expressed in tabular form, see Table 7.9. Here, the conditions present in each experiment at the Task level and the I/O level are shown. The nature of their influence on syntax is indicated. Since task structure, an approximation of topic assignment, was found to influence syntax usage (Experiment 2), it will be assumed that assignment of an object topic is responsible for the syntax preference in the observational study.

<table>
<thead>
<tr>
<th>Task level factor</th>
<th>Simple instructions: direct and literal (one object and one action). No topic assignment expected.</th>
<th>Object-based task structure</th>
<th>Action-based task structure</th>
<th>Assignment of object topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topic assignment</td>
<td>-ve effect</td>
<td>+ve effect</td>
<td>+ve effect</td>
<td>+ve effect</td>
</tr>
<tr>
<td>Task structure</td>
<td>no effect</td>
<td>no effect</td>
<td>no effect</td>
<td>no effect</td>
</tr>
<tr>
<td>Communications level factor</td>
<td>syntax</td>
<td>o-a</td>
<td>a-o</td>
<td>o-a</td>
</tr>
<tr>
<td>Syntax</td>
<td>-ve effect</td>
<td>no effect</td>
<td>no effect</td>
<td>no effect</td>
</tr>
<tr>
<td>I/O level factor</td>
<td>Input technique</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input technique</td>
<td>Command style input</td>
<td>Pick &amp; point style input</td>
<td>Pick &amp; point style input</td>
<td>Pick &amp; point style input</td>
</tr>
<tr>
<td>Command style input</td>
<td>-ve effect</td>
<td>no effect</td>
<td>no effect</td>
<td>no effect</td>
</tr>
<tr>
<td>Pick &amp; point style input</td>
<td>no effect</td>
<td>no effect</td>
<td>no effect</td>
<td>no effect</td>
</tr>
<tr>
<td>Pick &amp; point style input</td>
<td>(assumed)</td>
<td>no effect</td>
<td>no effect</td>
<td>no effect</td>
</tr>
</tbody>
</table>

Table 7.9
Summary Table expressing the findings of Experiment 1, Experiment 2 and the Observational Study

Secondly, the results will be expressed in more general terms. The results table depicts the conditions which influence Communications Level syntax. Inspection of the table enables the
net effect of any particular combination of conditions to be calculated. For example, if an interface with a command line style of interaction, required an object-action syntax and there was an action-based task structure, syntax performance would be expected to be very poor. Such a table is useful for determining the likely effect of particular sets of conditions. To validate the table fully, it would be necessary to conduct a series of empirical studies testing all sets of conditions under an identical experimental set-up. However, such an undertaking is beyond the scope of the work reported here.

The results table is shown in Table 7.10.

<table>
<thead>
<tr>
<th>Task level factor</th>
<th>Object-based task structure</th>
<th>Action-based task structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>topic assignment/</td>
<td>+ve effect</td>
<td>-ve effect</td>
</tr>
<tr>
<td>task structure</td>
<td>+ve effect</td>
<td>-ve effect</td>
</tr>
<tr>
<td>Communications</td>
<td>o-a</td>
<td>a-o</td>
</tr>
<tr>
<td>level factor</td>
<td>o-a</td>
<td>a-o</td>
</tr>
<tr>
<td>syntax</td>
<td>o-a</td>
<td>a-o</td>
</tr>
<tr>
<td>1/ O level factor</td>
<td>+ve effect</td>
<td>+ve effect</td>
</tr>
<tr>
<td>input technique</td>
<td>+ve effect</td>
<td>+ve effect</td>
</tr>
<tr>
<td>Command style</td>
<td>Command style input</td>
<td>Command style input</td>
</tr>
<tr>
<td>input</td>
<td>Pick &amp; point style input</td>
<td>Pick &amp; point style input</td>
</tr>
</tbody>
</table>

Table 7.10
General Results Table

7.4 Results of the Study and the Usability of Object-oriented Interfaces

The results of this study indicate that the object-oriented value on the dimension, syntax, is not inherently more usable, than the non-object-oriented value (action => object). Instead the ease-of-use of the two syntax structures is influenced by the type of task structure employed. When the task structure and the required syntax are consistent, i.e. the repeated item of the
task is placed in the initial position of the syntactic sequence, usability with respect to syntax is enhanced. This finding combined with that of the previous study, calls into question the assertion that an object-oriented interface is easier to use, in respect of syntax. Instead, it appears that ease-of-use can be determined by the manipulation of external conditions as the two syntax structures of interest seem to be equivalent.

8. Summary and Conclusions

The empirical activity reported in this chapter set out to determine whether the Task Level factor, task structure, influenced syntax usage at the Communications Level.

Results of previous studies (Chapters 4, 6) suggest that when performing office based tasks, users assign a topic for each particular task. The topic designates what the task is about, and functions as a superordinate goal for the task. For example, in a task where a memo has to be edited, printed and circulated, a user would think of the task as concerning the memo, the memo would be the topic for the task.

It is thought that the presence of a topic for a task might influence syntax usage. Users might tend towards expressing the topic first in a command sequence, or the reverse. However, the variable, topic assignment, cannot easily be manipulated experimentally, as the presence of an appropriate topic cannot be observed. Therefore, the variable, task structure, was manipulated and was assumed to approximate topic assignment. Task structure was varied by repeating either an object or an action over four sub-tasks, thus causing the repeated item to emulate a topic for the task.

The study had two parts. In the first part, subjects were assigned to one of two fixed syntax conditions (object => action; action => object). I/O Level factors were held constant across groups. All subjects used a Pick & Point interaction style as this has been found neutral with respect to syntax (Chapter 6). Subjects were asked to perform tasks which varied as to their task structure, i.e. some had an object-based task structure and some had an action-based task structure. The experimental hypothesis asserted that for an effect of task structure to be observed, the incidence of errors and partial errors would be greater and the latency to select the first command longer, on trials where the task structure and the syntax were inconsistent, (i.e. if the required syntax is object first and the task structure is action based).

It was found that under conditions in which the task structure varied frequently, task structure influenced syntax usage (in terms of errors & sub-optimal responses and time). The absence of an effect under block presentation conditions was discussed and an explanation based on the low task demands of this condition was proposed.
In the second part of the study, subjects were asked to use a variation of the device which had a free syntax, i.e. it would accept either of the syntax structures. Here, subjects performed tasks which had a task structure which was either consistent or inconsistent with their previous syntax experience (from the first session). In this session, the subjects' syntax preferences were recorded. It was expected that an effect of task structure would be identified if subjects modified their choice of syntax in accordance with the task structure. The data were in the expected direction, but statistical significance was not achieved.

The study enables the conclusion to be drawn that, under certain conditions, task structure does influence syntax usage such that the subjects tend to place the repeated item first in the command sequence. The effect occurs for both object-based and action-based task structures.

These findings were incorporated into the User Model so that it includes an ordering constraint at the Task Level which may conflict with the ordering constraint required (by the interface) at the Communications Level, thus resulting in a poorer performance.

The results were also used to extend the Summary Table set up in Chapter 6. This table now shows the effect of a Task level factor and an I/O level factor on syntax usage. Additionally, the results of the observational study (Chapter 4), were expressed in tabular form, where effects of object topic assignment were assumed to have similar effects to those of task structure.

To conclude, this study has provided some evidence that task structure influences syntax usage. The effect was found for both object-based and action-based task structures. The absence of an object advantage should be noted. The findings suggest that an object advantage would only be observed under conditions where objects dominate the task structure, or possibly, where objects are assigned to a topic role (there is no direct empirical support for this). Clearly further investigation is required to determine why an object advantage was observed in the observational study, when no such advantage has been found in the experimental studies. This question will be considered in the next chapter.
Chapter 8
The Ease-Of-Use of Object-oriented Interfaces: Two Views

1. Introduction

In this chapter, the results of the previous studies are considered.

Two contrasting views are proposed which concern the ease-of-use of object-oriented interfaces with respect to syntax. The Entity View asserts that the alternative syntax structures studied are equivalent with any performance differences being related to external conditions, while the Object Advantage View asserts that there is a performance advantage for an object first syntax due to inherent properties of objects. As yet, there is little evidence for the Object Advantage View.

Some candidate object properties which could be responsible for an object advantage are outlined and the evidence for them is discussed. From the information presented, it is thought plausible that some of the properties mentioned might contribute to an object advantage.

Next, the implications of the two views for the User Model are considered.

Finally, some suggestions for future work resulting from the consideration of the two views are presented.

2. Discrepancies between Studies

In Chapter 1, it was reported that object-oriented interfaces have a reputation for being easy to use. The study reported in Chapter 4 investigated this claim and it was found that an object-oriented interface (Apple Lisa) was easier to use than a non-object-oriented interface (IBM TopView) for a selection of office tasks. The subjects using the object-oriented interface completed more tasks successfully, made fewer serious errors, and required help less often. Additionally, it was found that the subjects using the non-object-oriented interface elected to use an object-oriented command syntax (object => action) even though this syntax was 'incorrect'. It was proposed that there was some advantage in having an object => action syntax. It was suggested that expressing the object first might be consistent with the user's representation of the task. If this were the case, it would be expected that a user's performance on an interface which required object => action syntax would be facilitated because fewer intermediary translations would be needed for a match between the user's representation of the task and the device's representation of the task. In
other words, an interface requiring object first syntax would have greater cognitive compatibility with the user.

However, subsequent studies (Chapters 6, 7) focusing on the usability of syntax structures under different I/O and Task Level conditions, failed to show any advantage of object => action syntax. There are two possible explanations why these studies did not show an advantage:

i) The use of the two syntax structures per se (object => action; action => object) is, in fact, equivalent. It is the effect of other factors such as task structure/topic which determine any object (or action) advantage.

ii) Objects have properties which are responsible for the enhanced consistency between device and user representations when an object => action syntax is used, but these properties were not investigated in the syntax studies.

Each of these explanations corresponds with a particular view of the usability of object-oriented interfaces with respect to syntax. These views are described below.

2.1 The Entity View

The Entity View asserts that an object first syntax and an action first syntax are equivalent, all other variables being equal.

In terms of this view the most important factor for ease-of-use is that an interface has explicit entities. Entities correspond with the task elements and can take the value 'object' or the value 'action'. No special properties are believed to exist for entities by virtue of their being an object or an action.

2.2 The Object Advantage View

The Object Advantage View asserts that an object first syntax is more consistent with a user's representation of a task, due to properties particular to objects.

In terms of this view, objects can be differentiated from actions by the properties which make them more suitable to appear first in a syntactic construction. This view is akin to the notion that objects are more 'naturally' understood than actions.

The empirical syntax studies favour the first view, the 'Entity View', while the observational study could be explained in terms of either view. The advantage of object => action syntax found in the observational study could be attributed entirely to a topic or task structure effect, in which
case an equivalent action advantage would be expected under conditions where an action was recruited as the topic, or was more prevalent in the task structure. Alternatively, the object advantage could be a result of a combination of topic and other (as yet unspecified) object specific effects. The commonly held belief that object-oriented interfaces are easy to use implies that the use of objects per se enhances compatibility between the user and the device. With regard to syntax, no convincing support has been found for such a notion.

As yet no consideration has been given to the possible properties of objects which might afford them an advantage over actions. In the next section, candidate properties will be proposed and evidence for them discussed.

3. Possible Factors Responsible for an Object Advantage

Before judgments can be made concerning the validity of either of the above views, more information concerning the second, Object Advantage View, is needed. In particular, the possible object specific properties which have consequences for syntactic position, need investigation. Three properties have been identified: one is related to an interface property which has been determined from the results of the observational study; the second has been determined from consideration of the psychological literature; and the third stems from the findings of word association studies. Each of the properties will be discussed in the following sections.

3.1 Hiding

'Hiding' refers to an interface property which has been identified from a consideration of the findings of the observational study. It is a useful expression because potential differences between objects and actions (as represented by the device) can be explored in relation to it.

Some evidence for an object priority was found in the observational study. Some of the dimensions used to distinguish object-oriented interfaces were found to have an influence on subjects' performance. These dimensions were syntax; concealing the mechanics of the programme from the user; object to action ratios (though not in the expected direction) and the representation of objects. It was noted that subjects using the non-object-oriented interface had difficulties with navigation and determining the location of items. These difficulties were attributed to the presentation style of TopView, where objects are represented as items in lists.

When the results of the study are viewed in a general fashion, it can be seen that the advantages of the object-oriented interface (in terms of its ease-of-use) are related to the extent that it hides its complexities of workings from the user. Instead of being forced to learn computer specific
processes and terms, the user can interact in ways which are meaningful in terms of the domain. To some extent, hiding could be seen as the function of an end-user interface i.e. that it allows communication between the user and the computer without expecting the user to work in binary. However, in a more sophisticated sense, hiding can be seen to be a property of interfaces which is possessed differentially. Some interfaces hide more of the complexities of their workings than others. It seems that hiding is a superordinate interface property which can be broken down into a number of specific device features. For instance, hiding seems particularly related to the dimensions, 'concealing the mechanics of the programme from the user' and 'representation of task entities'.

Hiding has a potentially interesting connection with the use of objects in interfaces which will be discussed later.

First, some examples of hiding will be described. The examples are derived from a comparison between the two systems used in the observational study (TopView and Lisa).

i) Lisa allows a user to access a document and begin editing it immediately, irrespective of its type (textual; graphical; numeric etc.). The user is not required to signal that a specific application programme should be called (e.g. LisaWrite; LisaDraw; LisaCalc etc.). This automatic invocation protects the user from interacting with different application programmes; instead s/he works with a document. The user need not be aware that the different data types are dealt with by separate programmes. By contrast, in TopView, the user must explicitly switch to the appropriate application programme to edit a particular type of document.

ii) Lisa automatically reads an object's identity and address when that object is selected. After selection, there is no need for the user to specify any further information about the object. In TopView, when using the Personal Editor programme, the user is required to type in the file address of the to-be-edited file. Often, in command line style interfaces, the user has to type in filenames and addresses as parameters to commands, even if the file names have just been listed. Direct selection is not possible.

iii) Lisa hides the use of modes by presenting the user with valid options only. An alternative strategy is to signal the mode and then limit the user's selection by disallowing unavailable options. Here, the existence of modes is not hidden.

More information concerning the construct,'hiding', is given in the following sections: first its relationship with other similar constructs found in the HCI literature is considered; then advantages of hiding are discussed and finally possible differences between objects and actions with respect to hiding are identified.

219
3.1.1 The Relationship between the Notion of Hiding and other Concepts in the Literature

There are a number of concepts used in the HCI literature to describe desirable interface properties which bear some relation to the notion of hiding. Several of these concepts will be described briefly and their relationship with hiding considered.

3.1.1.1 Related concepts in the HCI Literature

*Transparency*

Transparency refers to the degree to which an interface can support the performance of a task unobtrusively (i.e. reducing the need for computer specific learning for successful completion of a task). Rutkowski (1982) claimed in his recommendations for easy-to-use interfaces that:

"In an ideal situation the relationship of user and tool approaches one of transparency. The user is able to apply intellect directly to the task; the tool itself seems to disappear".

Holtzblatt, Jones and Good (1988) investigated transparency as a usability concept. They observed users in the field in order to document disruptions in the work flow. They see disruptions to users' work as evidence for low transparency:

"Systems can disrupt users by fragmenting the task into elements which do not match the user's view of the task. When users are forced to attend to the application or are blocked from completing a task by the application, we see disruption in the work flow".

*Readiness-to-hand*

'Readiness-to-hand' is a concept adopted from Heidegger's work which has been applied to computer interfaces by Winograd and Flores (1986). Heidegger argued that objects and properties are not inherent in the world, they only become apparent when there is a 'breaking down' in which they become 'present-at-hand'. An example, given by Winograd and Flores, says that when someone uses a hammer to drive in a nail, the hammer as such does not exist for that person. It is part of the background of 'readiness-to-hand' that is taken for granted without explicit recognition or identification as an object. It is part of the hammerer's world, but is not present any more than are the tendons of the hammerer's arm. The hammer only presents itself (or is acknowledged explicitly) when there is some kind of 'breaking down' or unreadiness-to-hand. Its 'hammeriness' emerges if it breaks or slips from grasp or mars the wood, or if on deciding to drive in a nail, the user cannot find a hammer.
Winograd and Flores assert that all successful tools should approach 'readiness-to-hand' in that the user should be unaware of the actual particulars of a tool during its unhampered use. They comment:

"A successful word processing device lets a person operate on the words and paragraphs involved without being aware of formulating and giving commands".

First Personness

First personness is a concept introduced by Laurel (1986). Influenced by dramatic theory, she begins from the Aristolean premise, "that the end cause of a thing is the function that it is intended to serve; that is, what it is supposed to do". If applied to interactive computer programmes, it seems that their main function is to enable the user to achieve some goal such as compose and format a document (word-processor); access information (database); play a game etc. Frequently, this is not the case as the user is required to acquire a secondary goal of learning to use the computer system, however, Laurel proposes that this can be avoided by ensuring engagement at the level of the interface. It must be possible for a person to become engaged in the unique context of the world represented by the computer. Laurel also writes that in order to achieve the desired experience during interaction, the user should be able to act within a mimetic context. The contrast being the situation where the interface is presented as being an intermediary. To foster this engagement, the interface must possess the desirable quality termed, 'first personness'. For first personness it must be possible for the user to operate in the 'world' (a 'you-are-there' experience); this form of interaction contrasts with the second personness of the command style interface where the computer prompts and the user issues commands.

E.g.

COMPUTER PROMPT: Enter filename:
USER COMMAND: c:\data\exp.

Laurel cites an illustratory example of first personness. In the Atari game, 'Pole Position', the user 'drives' a simulated race car down a track. The speed of the car can be controlled by a foot pedal, rather like a car accelerator. The first personness of such an interaction would be undermined if, for example, the user was required to specify the required speed in a numerical representation via the keyboard. Similarly, there would be a reduction in first personness if the effect of the motion of the footpedal was represented as numerical information on screen rather than the animated representation of the car getting faster and slower.
Such an example might suggest that the essence of first personness lies in input-output methods or devices. This is an over-simplification, Laurel states that instead it relies on a thorough understanding of what the user wants to achieve and the best (natural) method of supporting the advancement of the user's goal and providing feedback about the process.

Direct Engagement

In their analysis of direct manipulation interfaces, Hutchins, Hollan and Norman (1985, 1986), identify that the class of direct engagement interfaces creates a feeling of 'directness' for the user. They propose that this directness can be divided into two components: distance and engagement. The first of these, distance, can be further sub-divided into semantic and articulatory distance. These refer respectively, to the mapping of the command words/entities of the system onto the goals and intentions of the user and the mapping between the means of representation and what it is actually meant to represent.

The second aspect of directness is of particular interest here. It bears a resemblance to the notion of engagement put forward by Laurel. It is said to occur when a user experiences direct interaction with the objects in a domain. There is a feeling of direct involvement with a world of objects (where the interface is built on the 'model world' metaphor, Hutchins (1986)) rather than of communication with an intermediary ('conversation' metaphor). Hutchins et al. believe the feeling of direct engagement to be of critical importance. However, they admit to knowing very little about the actual requirements for producing it.

3.1.1.2 The Relationship between the above Concepts and Hiding

It would seem that all the above concepts are similar, even though they have been proposed in the context of different technologies.

All of the above seem to impact upon the interface as a whole i.e. they refer to a global property, they describe the 'feel' of an interface (it may be an interface which engages the user or not). While all of the authors can give examples at a lower level to demonstrate why a particular interface can said to be 'engaging'; 'high in first personness'; 'transparent' etc. They do not provide a rationale for determining the features responsible for 'transparency', 'first personness' etc. Any comparisons between different implementation styles are reported anecdotally. No criteria are supplied for judging interfaces with respect to the presence of such features.

Hiding, while being in accord with the apparent meaning of the above concepts, is slightly
different, in that it can be applied at a lower level. It requires an acknowledgement of what exactly has been hidden, therefore direct (and numerical) comparisons can be made between systems. Hiding also includes an opportunity to reflect on its converse, 'exposing', which under some conditions may be advantageous. 'Exposing' occurs when the user is given information about particular system states. However, it has not been established whether the degree of 'hiding' is correlated with increased ease of use.

Despite some differences, the underlying reason for the desirability of hiding and the above concepts appears similar, i.e. a reduction in the complexity for the user.

Hutchins, Hollan & Norman write:

"The reduction in the cognitive load of mentally maintaining relevant information and the form of the interaction contribute to the feeling of engagement".

Reduction in complexity can occur at a number of different levels in an interaction. Some possibilities are described in the next section.

3.1.2 The Potential Advantages of Hiding

It remains to be determined whether hiding is advantageous and if so, to what extent and under what conditions. It has been mentioned that hiding, in common with the other concepts, probably contributes to ease of use by reducing complexity. Two ways in which complexity might be reduced are given below:

i) Hiding makes an interaction less complex because it reduces the number of steps that a user has to perform. E.g. if the typing of complex path-names is avoided then there is a lower potential for error (either of understanding or simply of finger-slip).

ii) Hiding operates to reduce complexity, in a more general sense, as it reduces the amount of computer-specific learning. If the device more nearly approximates the application domain, then the user can concentrate on performing tasks within the domain, without translation between the non-computer supported and the computer supported versions of the task. This form of complexity reduction is similar to the advantages quoted for the other similar concepts in the literature.

The second means of complexity reduction might give the impression that for an interface to be 'good' at supporting tasks, it should provide a very faithful simulation of the present world, where those tasks are performed. For example, for natural interaction with representations of files, the user should be provided with a representation of a filing cabinet which mimics the behaviour of its counterpart in the real world etc. However, this is not the case. Johnson (1987)
pointed out some of the pitfalls in expecting a faithful metaphor to provide the best computer solution:

i) there may be difficulties associated with the present method of achieving a goal when it is represented on screen (since user dexterity in manipulating screen objects is not high);

ii) the computer implementation may be restricted to the limitations currently found in the physical world.

An example, given by Jefferson, is that of wanting to read some text. In the physical world, the text may be contained in a book which has pages of finite length. Therefore, to locate a particular section of text, which does not appear on the first page, the reader will have to turn over a page. This action is fairly readily accomplished in the physical world; however this might not be the case in a direct computer simulation where the user might have to select (via mouse) the corner of the to-be-turned page and keeping it selected, make a sweeping movement to the other side of the screen (with the whole sequence being animated). In the first place, such a task would place unreasonable demands on the user's dexterity and secondly it misses the point of what the user is trying to achieve i.e. read some text (also it is wasteful in terms of hardware/software development). Better is the case where the spirit or essence of the user's interactions with objects is simulated instead of the actual details of it.

3.1.3 Hiding and Objects

It has been suggested that one of the advantages of object-oriented interfaces is that they embody a reasonably high degree of hiding. It should be considered whether employing an object-based structure automatically increases the level of hiding or whether examples of object-oriented interfaces coincidentally have both objects and hiding.

In the observed contrast between Lisa and TopView, it seemed that the use of objects did enhance hiding in the case of the Lisa. An object (as represented by the device) has two properties which seem pertinent to hiding these are, 'Identity' and 'Location' (or 'physical representation').

Below are some examples which illustrate that Lisa has a higher degree of hiding than TopView, by virtue of its objects.

i) Editing (as described above)

Lisa
The Lisa does not require the user to be aware of the different application programmes or to use them explicitly. Lisa appears to deal with objects as items with a particular identity and
location, such that selection from the screen is sufficient for the device to retrieve any details necessary for storage and modification.

*TopView*

The user has to type in the name (identity) and address (location) of the item to be edited, even though the item can be seen in the DOS Services window.

ii) Changing directory (returning up the hierarchy):

*Lisa*

The user can either resize the current window and then select the window or icon of the folder containing the current item or the user can close the present window which results in an automatic return to the next container up. Here, the representation (either as an icon or a window) of a previous folder/container <OBJECT> is not lost when a new one is selected. This serves to 'hide' the hierarchical naming convention and relies upon the manipulation of physical entities instead.

*TopView*

The user must re-type or edit the address which appears at the top of the present directory window and press <Return>. E.g. c:\TopView\Letters\*. becomes c:\TopView\*.

The new directory listing immediately overwrites the previous one. In TopView, the user is expected to work in terms of the hierarchical file structure, the notion of 'containing' is not well supported and so the user must translate from 'is contained in' to 'is one step down the hierarchy from' in order to make the appropriate changes to the path-name. Even learning constructs such as 'path-name' is not directly related to the user's task goals. Additionally, in TopView, any directory that is opened results in the same screen display (except for the particulars of the address and the filenames of the contents), so new directories prevent the user from seeing previous ones.

iii) Moving a file from one container (folder or sub-directory) to another:

*Lisa*

The user must select the icon of the item and drag it to its new 'home'.

*TopView*

The user must copy the item to its new destination (which involves typing in a path-name string) and then erase the original item from its original place. There is no explicit MOVE option in DOS. Therefore, the user has to derive it from the available options: a two-stage function of COPY then ERASE.
This operation might be only indirectly related to objects and hiding. It is assumed that the presence of the object in the Lisa user interface, obliges the computer to have all common valid actions associated with the object present. Thus, if one has a file which can be placed in containers it must be possible to move it between them without copying. The difference between the two systems in this respect seems to exist at the hardware level. The PC chip would appear not to have 'MOVE' a file (i.e. alter its directory [ownership] details) in its function instruction set; whereas the Apple machines do. However, this could be rectified at the interface by the PC 'hiding' the limitation from its users. One method would be to supply a ready-made 'MOVE' option that simply concatenated 'COPY' and 'ERASE' for the user, in the manner of a macro-command.

There is some support for the idea that structuring an interface in terms of objects can increase the incidence of hiding. It would be interesting to attempt a demonstration that objects increase hiding by devising two interfaces of equivalent functionality, one based on initial object representations and the other on initial action representations, and comparing the instances of hiding. Having identified one possible way in which objects per se might differ from actions, the next step is to consider other differences between actions and objects.

3.2 Differences in Representations for Knowledge about Objects and Actions: Evidence from the Psychological Literature.

It is possible that there are psychological differences in people's perceptions concerning objects and concerning actions. In particular, it is of concern whether knowledge concerning action concepts and knowledge concerning object concepts is represented in the same way. If differences exist, these may influence the way in which users interact with interfaces which represent objects and actions. Here, sources of psychological literature will be reviewed to determine the extent of evidence for differences.

The amount of direct evidence concerning differences between the representation systems for objects and actions is small: few experimenters have performed a direct comparison. However, some of the work that has been carried out in the areas of semantic memory and word association is of interest. Relevant research from each of these areas will be summarized below.

Note that in each of the following sections the terms, 'object', and, 'action', are taken to be equivalent to the referents of common nouns and verbs; i.e. concrete nouns and verbs of action or indication of state.
3.2.1 Analytic Research in Semantic Memory.

Unfortunately, a large proportion of the work in the area of semantic memory has centred around object concepts and their categorisation and has not considered the representation of action concepts. These are often dealt with in conjunction with object concepts in theories of episodic memory e.g. Schema Theory, (Rumelhart & Ortony, 1977).

However, two sorts of evidence from the semantic memory work are of interest: assertions that there are different organizations in memory for object and action concepts (and models of the organizations) and information about the relationship between object and action meanings (i.e. whether they are independent or inter-dependent).

There is considerable support for the idea that the representation systems for objects and actions differ. Even if similar representation mechanisms are posited e.g. propositional representations, then it is acknowledged that the form or content of these propositions will differ for objects and actions.

One such theory is Kintsch's theory of semantic memory (1972, 1974). He identifies a need for two representational systems (though both are propositional). The system for nouns is based on propositions of the form (B, A) where A and B are both nouns, i.e. a set-inclusion hierarchy 'cat is an animal' etc. The notion of nouns forming a hierarchy based on the relation 'IS-A' is common to most work on semantic memory. When he considers verbs, Kintsch points out that, 'the methods used for the classification of nouns cannot be applied in quite the same way . . . because verbs must be classified together with the arguments that they take'. So he proposes a system in which the propositions specify the likely arguments and the case of these arguments (AGENT; INSTRUMENT; OBJECT etc.).

Kintsch proposes:

i) that there are different representations for nouns and verbs;

ii) that verbs rely on nouns to specify their meaning.

Kintsch defends his separation of noun and verb systems by citing theoretical and empirical evidence for a noun-priority in the lexicon. The evidence he cites is as follows:

i) Chomsky (1965) when substituting words into phrase-markers, first inserts the nouns by means of context-free rules, and then selects verbs and adjectives in terms of nouns - he shows that the converse method leads to unacceptable complications of the grammar.

ii) Marshall & Newcombe (1966) report the case of a dyslexic patient who could read nouns quite well, but who performed poorly on adjectives and verbs.
iii) Weigl & Bierwisch (1970) report that a dyslexic patient is often able to read a sentence if s/he is told what the sentence is about using noun substitutions; but is unable to do the task if verb substitutions are made.

These findings are taken to suggest that nouns are more accessible than verbs.

3.2.2 Empirical Work in Semantic Memory Organization.

There are two studies which are pertinent to the issue of the nature of the representation systems. The first was performed by Huttenlocher & Lui (1979) who set out to explore the semantic organization of simple nouns and verbs (those referring to concrete objects and actions) and its development. Three different sorts of memory tasks were used: release from proactive inhibition; free recall and ordered recall. The results showed effects of semantic relatedness on short-term memory for both nouns and verbs - but the effects differed.

They argue that results for nouns indicate that these are represented in memory as a multi-level hierarchy whereas verbs are more suited to a matrix-like organization. They propose that concrete nouns encode categories of objects which exist independently of their actions whereas verbs encode categories of actions and states which require objects - they encode conceptually dependent categories. For example, the act of 'eating' involves an animate creature and an ingestible substance. Conversely, although there is a strong association, an object such as an apple exists independently of acts of eating. An asymmetric relation is therefore assumed to hold between nouns and verbs.

Some models of the lexicon have been proposed which focus on the verb (e.g. Norman & Rumelhart 1975); so that object categories are defined in terms of their actions or functions. However, these models are based on the representation of sentence meanings. Huttenlocher & Lui argue that it is "precisely because verbs are conceptually elements that they are central in sentential meaning, relating nouns to one another as their arguments . . .".

Clearly Huttenlocher & Lui advocate:

i) that there are different representations for nouns and verbs;

ii) that verbs are conceptually dependent upon nouns for their meaning.

The second study, by Graesser, Hopkinson and Schmid (1987), investigated the results of Huttenlocher & Lui: that sets of nouns tend to be interrelated according to a hierarchical organization, whereas sets of verbs are better suited to a matrix organisation. In two experiments, subjects performed a set of three tasks on sets of nouns, action verbs and event verbs. The first was an unconstrained sort task, followed by a binary sort task and then a similarity weighting task.
The binary sort task was expected to yield 'hierarchical organization scores', whereas the similarity weightings would provide 'matrix organization scores'. The extent to which these scores could predict the clustering of words in the unconstrained sort task was assessed. The results confirmed the hypothesis. Again, it was argued that noun and verb meanings are organised differently in semantic memory.

In summary, the work in semantic memory seems to suggest that there is evidence for believing that there are different representation systems for object and action meanings. It seems probable that the system for nouns assumes a hierarchical structure, while the verb system is more matrix-like. However, the evidence as to the relationship between object and action meanings (whether the interdependency is in the direction object => action or action => object) is equivocal. Clearly some models rely on actions being defined through objects, whereas for others the converse is true. It must be noted that the models have often been devised for different purposes e.g. representing single word meanings versus representing sentence comprehension. Empirical evidence to support either arrangement is sparse.

3.2.3 Findings from Word Association Experiments.

While not a direct source of evidence for the nature of representations of knowledge about objects and actions, word association experiments might provide insight upon the sorts of interrelationships which exist between types of words. Cramer (1968) provides a review of a vast number of word association experiments which were performed between 1950 and 1965 and provides a summary of the main points pertaining to the effect of the part of speech of the stimulus.

It is reported that:

i) the reaction time for producing a response is longer for verb stimuli than for noun stimuli;
ii) fewer responses are available for nouns than for verbs;
iii) the number of different responses given is greater for verbs than for nouns;
iv) the primary response is stronger for nouns than for verbs
v) the response commonality (judged by the strength of the first three responses) is greater for nouns than for verbs.

A number of researchers have distinguished between paradigmatic responses, i.e. those where the response word could be fitted into a sentence frame in place of the stimulus word (e.g. 'He was a good/bad boy', here, 'good' and 'bad' are paradigmatic) and syntagmatic responses, i.e. those where the response word is a successive word of a sentence frame containing the stimulus word (e.g. 'It is extremely difficult', here 'extremely' and 'difficult' would be syntagmatic). It has been
found that this variable differs with part of speech:

i) nouns elicit a higher number of paradigmatic responses than verbs;
ii) the number of syntagmatic responses is greater for verbs than for nouns.

These latter findings suggest that it is more likely that verb stimuli elicit responses from other grammatical classes (non-verbs) than nouns which seem to elicit other nouns. Although an analysis of grammatical class of response is required, there is certainly a possibility that verbs may elicit more nouns than nouns do verbs - this may be indicative of some interdependency of meaning.

3.2.4 Summary

Clearly, there is evidence both theoretical and empirical that there are different representation systems for objects and actions. Also, there seems to be support for the idea that meanings for objects and actions are not independent; however the direction of the potential inter-dependency is undecided. Some theorists propose that actions depend upon their associated objects for their meaning/context, but other theorists propose the converse. There seems to be little conclusive empirical support for either position.

3.3 Interrelationships between Nouns and Verbs

3.3.1 Evidence from the Associative Thesaurus.

It was thought that the Associative Thesaurus was a potential source for further, more specific information about possible noun-verb interrelationships or interdependencies. The Associative Thesaurus was compiled by a research team at the MRC Speech and Communication Unit, Edinburgh, between 1968 and 1971 (see Kiss et al., 1974). It contains associative responses for 8400 verbal units and words of English. The data were collected from a series of free association tests from British undergraduates. Approximately 2000 words were selected to be used as initial stimuli. The responses to these initial stimuli were then used as stimuli in subsequent tests so that an associative network was built up.

The stimulus words were printed on sheets and subjects were asked to write beside each one, 'the first word that it made him/her think of'. Subjects were asked to work as quickly as possible. The stimulus sheets were arranged, so that each subject had to respond to 100 words, and 100 responses were obtained for each stimulus.

In the thesaurus, there are two sorts of listing:
i) each stimulus word is listed with all the responses elicited and the frequency of each response.

ii) each word that was given as a response is listed with the stimulus words which evoked it and the frequency of its occurrence.

Thus, the index of strength of association seems to be the most frequently occurring response to a stimulus (out of a possible score of 100).

3.3.1.1 Consulting the Thesaurus

The issue of interest was to discover whether there is an asymmetry between the representations of meaning for objects and actions. It has been speculated that there might be an interdependency such that action concepts rely on object concepts for their context. A useful exercise would be to consult the thesaurus to find out whether the action concepts associated with particular objects are the same as the object concepts associated with particular actions.

A list of 15 common object and action words was selected. This prior selection was intended to reduce possible bias concerning the organization of the thesaurus. The entries in the thesaurus for each of these words were then examined.

The words were:

<table>
<thead>
<tr>
<th>OBJECTS</th>
<th>ACTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple</td>
<td>Close</td>
</tr>
<tr>
<td>Coffee</td>
<td>Draw</td>
</tr>
<tr>
<td>Cup</td>
<td>Eat</td>
</tr>
<tr>
<td>Chair</td>
<td>Buy</td>
</tr>
<tr>
<td>Book</td>
<td>Ask</td>
</tr>
<tr>
<td>Cat</td>
<td>Drink</td>
</tr>
<tr>
<td>Coat</td>
<td>Hear</td>
</tr>
<tr>
<td>Door</td>
<td>Look (at)</td>
</tr>
<tr>
<td>Bed</td>
<td>Open</td>
</tr>
<tr>
<td>Bag</td>
<td>Make</td>
</tr>
<tr>
<td>Hair</td>
<td>Bake</td>
</tr>
<tr>
<td>House</td>
<td>Pick (up)</td>
</tr>
<tr>
<td>Flower</td>
<td>Read</td>
</tr>
<tr>
<td>Glove</td>
<td>Wear</td>
</tr>
<tr>
<td>Letter</td>
<td>Hit</td>
</tr>
</tbody>
</table>
The entries were examined to determine the most common responses to each entry as well as to find out the proportion and content of verb responses to nouns and the proportion and content of noun responses to verbs.

3.3.1.2 Problems Encountered

During the examination of the thesaurus a number of problems were encountered which highlighted the limitations in the usefulness of the Thesaurus for the purposes stated above. These problems are described briefly.

i) the nature of the task demands

Subjects were only required to give one response and there was no constraint placed on the type of response given i.e. the relationship between stimulus and response was open. A number of types of response were given e.g. synonyms; antonyms; homonyms; rhyming words; other members of a class etc. It was noticeable that a fairly high proportion of responses were of the 'completing a phrase' type. Providing this type of response is rather like playing a word game.

Examples are:

CLOCK - tower
- work

FLOWER - power

HIT - record
- parade

While it is interesting that associations of this type were 'the strongest' for many subjects; they are not very informative for the purposes here. Often they serve to transform the stimulus word into another concept.

Further, there is no guarantee that the word that is written down is actually the one with the strongest association. It is feasible that in the time required to provide a written response, subjects could generate a number of responses and their response represents the result of some type of selection procedure, not necessarily consistent with strength of response.

ii) considering grammatical class
a) subjects were not informed of the intended grammatical class of a stimulus word. In English, there is scope for a considerable degree of ambiguity. From the responses generated, it is clear that different subjects made different assumptions concerning the class of the stimulus.

Example:

PICK - choose <- this response indicates that the subject was assuming the stimulus as a verb.

- axe <- this response indicates that the subject was taking the stimulus to be a noun.

b) in inspecting the responses to a stimulus it was not always clear whether the response was intended to be a verb or a noun - this left the grammatical class of the response open to interpretation.

Example:

LOOK - glance <- this could equally be assumed to mean 'a glance' [noun] or 'to glance' [verb].

Thus, rating responses with respect to grammatical class is an uncertain process. Also, it is not clear that all of the subjects that responded with a particular word intended it to have the same meaning. If 15 people responded with 'glance'; 3 could have intended it as a noun and 12 as a verb, but such details cannot be distinguished from the Thesaurus.

c) a further problem concerned rating responses with respect to grammatical class: sometimes an apparently straightforward response was found to be open to a double meaning.

Consider the stimulus 'CHAIR' - some subjects responded with the word 'LIFT'; alone this would seem to be a verb response ('to lift'); but when read as a phrase 'CHAIR-LIFT' an object is clearly intended - however 'LIFT' alone is not really sufficient to denote the object.

3.3.1.3 General Findings

Despite the problems described above the findings of the consultation were:

i) a similar pattern was obtained as that found in reviews of word association experiments in general;
ii) More particularly, an examination of the grammatical class of responses shows that:

- for nouns the responses seem to be mostly other nouns and adjectives (e.g. the names of other members of the class; the superordinate class name and properties of the referent)

  e.g.  APPLE - fruit
       - banana
       - pear
       - green

- for verbs the responses tended to be other verbs; nouns upon which the verb might act and adverbs & conjunctions to make up common phrases e.g. PICK - up etc.

  e.g.  OPEN - shut
        - door
        - case

  DRINK - water
        - beer
        - swallow

From the sample of common nouns and verbs examined it was found that the nouns yielded at least 60% of other noun responses (in most cases over 80%), whereas the verbs only yielded 40% of other verb responses.

c) Some words were found to have a particularly strong noun-verb association that worked symmetrically e.g.

**BOOK - READ**

'BOOK' as a stimulus yields 'READ' in 18% of cases.
'READ' as a stimulus yields 'BOOK' in 31% of cases.

It was more common for a verb to elicit a noun response than it was for that noun to elicit the verb. Often the most salient response for a verb would elicit that verb, but only at a very low frequency.

Sometimes it was difficult to tell whether real asymmetry existed or not e.g.

'SEND' as a stimulus elicits 'LETTER' in 15% of cases.
'LETTER' as a stimulus elicits 'POST' in 19% of cases and 'SENT' in 1% of cases.

Here, 'POST' could have an equivalent meaning to 'SEND' as in 'to post a letter' in which case reasonable symmetry could be assumed; however, it could also denote the more general concept 'the Post' in which case it would have a generic class type relation to the stimulus. Here the single, unconstrained response paradigm becomes unsuitable for the present purposes.

3.3.1.4 Conclusion

While consulting the thesaurus seemed a potentially useful exercise, in practice a number of factors contributed to render the results uncertain i.e. the difficulty in ascertaining the grammatical class and intended meaning of responses; the possible non-homogeneity of meaning of any particular response across subjects giving that response etc.

However, even this cursory examination lends support to the claim that nouns are more likely to be implicated in the representation of meaning of verbs than the converse. Such a conclusion can only be drawn, however, if the 'strength' of association in a free association test can be taken as an indication of the way that semantic memory is organised.

3.3.2 An Exploratory Study Addressing Object Action Asymmetries

An exploratory study was designed to investigate the possible asymmetry between objects and actions. The study was similar in approach to the word association studies reported above. However, the possible responses were limited to ensure that the relationships of interest were sampled. In other words, subjects were asked to indicate actions in response to object stimuli and vice versa. It was of interest whether the responses for the object stimuli would differ, in a systematic way, from the responses for action stimuli.

3.3.2.1 Details of the Experimental Testbed

The experimental testbed for the study was programmed in HyperCard.

Subjects were given instructions on the screen. They were instructed that they would have to pair common objects and common actions. It was explained that for objects, they should select actions which could be performed on the object. For example, if the word were 'APPLE' then appropriate pairs might be 'EAT' or 'PEEL', but 'DRINK' or 'BOUNCE' would not be appropriate. Then for actions, they should select the objects on which they could be performed. For example, if the action were 'STIR', then the pairs, 'SOUP' or 'COFFEE' would be appropriate whereas 'TOAST' and 'TABLE' would not. Subjects were given a selection of responses and they were asked to select
the appropriate ones by clicking over the word with the mouse. On selection, a '✓' would appear next to the item. Subjects were warned that they would not be able to de-select an item. Following the selection(s) of relevant items, subjects were told to click on a 'FINISH' button to initiate presentation of the next stimulus word. Prior to the test session, subjects were given two practice trials, identical in form to the test sessions, but using kitchen-related objects and actions as stimuli.

In the test session, eighteen common office objects and actions were used:

LETTER DIARY WASTEBIN TELEPHONE FOLDER DIRECTORY CALCULATOR CARD-INDEX GRAPH DIAGRAM CLOCK FILING-CABINET IN-TRAY MEMO ADDRESS BOOK CALENDAR REPORT NOTEPAD
EDIT EMPTY SEND SORT LOOK IN TYPE COPY SWITCH ON PICK UP THROW AWAY REPLY SET ENLARGE CONSULT DIAL RELABEL DRAW FILE

Subjects were asked to think of these objects and actions in the context of performing office tasks, e.g. in the case of a DIARY, while the action, THROW AWAY, could be performed on it, it would be unlikely that this would happen as a regular office task.

The screen layout for the test session always took the same form, as shown in Figure 8.1.

![Screen Layout for Exploratory Study](image)

Figure 8.1
Screen Layout for Exploratory Study
All subjects were given a mixture of half object and half action stimuli. Each subject had eighteen stimuli. There were four variants to ensure that all of the 36 possible stimulus words were sampled equally. The items which appeared in the response set were presented in a randomly determined order on each trial.

The experimental testbed also included a facility for recording all the responses for each stimulus word. Ordering information, i.e. the order in which the responses were selected, was not found to be useful as all subjects adopted the strategy of working down the list of response options and making a decision about each one in turn.

3.3.2.2 Design

The study had one independent variable, stimulus word type (object or action) which was a within subjects variable. The dependent variable was the set of response words selected. Each of the four interface variants, as mentioned above, was used by an equal number of subjects.

Twenty subjects participated in the study. All subjects were students or research staff from University College London. Participation was voluntary and each subject was paid £1.00 for a session lasting 1/2 hour. Thirteen of the subjects were male and seven were female. The mean age was 30.4 years (range 22-44 years). All, but one, of the subjects had used a mouse before.

3.3.2.3 Results

The data were recorded in the form of the set of responses for each stimulus word. A matrix was produced comparing the responses for each object stimulus with the responses for each action stimulus. Scores were assigned to responses to indicate the number of subjects who had selected that response for a given stimulus. A score of ten was assigned if all ten subjects receiving the stimulus gave a particular response, e.g. for the word, EMPTY, all ten subjects selected WASTEBIN, so a score of 10 was assigned to the cell [EMPTY-WASTEBIN]. A comparison can then be made with the cell [WASTEBIN-EMPTY], to find out how many subjects responded to the stimulus WASTEBIN with the response EMPTY. Comparing the two scores gives an indication of the symmetry or asymmetry of response. (The matrix is presented in Appendix J).

Each pair of complementary cells was compared and it was found that their scores were surprisingly similar. In 75% of the comparisons, the scores of the complementary cells (object-action vs action-object) were the same or they differed by a single subject. No systematic patterns were found in the comparisons where the two scores differed.
It had been expected that a difference would have been observed if one stimuli type had elicited a profile of very high or very low scores, i.e. either all of the subjects, or none, selected a response and the other type had elicited a profile of intermediate scores, i.e. where only half of the subjects selected a response. This effect was not found.

3.3.2.4 Conclusions

The exploratory study did not reveal any evidence for an asymmetry between the pairing of given object and action words. In fact, the responses for the two stimuli types were surprisingly similar. It is possible that effects would only be observed under very large sample sizes (there were 200 subjects used to gather the Associative Thesaurus data). In addition, it is possible that the design of the study reduced the likelihood of effects. Here, subjects used recognition memory to respond - all the potential responses were given. Asymmetrical effects may only be present when subjects are required to recall responses from memory.

3.4 General Summary Concerning Object Specific Properties

Three characteristics pertinent to object action differences have been described and the evidence for them discussed. The way in which these characteristics might contribute to interface usability is briefly considered.

The ability of objects to promote hiding at the interface has an obvious connection with interface usability. It would be interesting to investigate this property by comparing devices biased towards objects with those based on actions, while controlling the degree of hiding. The question as to whether there is a true relationship between object bias and degree of hiding could be addressed. The second property of objects, that they are represented differently in semantic memory, has a less direct connection with usability. However, differences in representation may impact the relative accessibility of objects and actions for the user, which might suggest that the most accessible item is available to the user first. Also, it is possible that there are consequences for performance based on the mapping between the device's representation of objects and actions and the way that the user stores knowledge concerning the two types of concept. However, an empirical test of such a conjecture would not be straightforward. Potential interdependencies between the meanings of objects and actions could have consequences for user performance. If, for example, actions are dependent on objects for their meaning, then a device which restricts its initial presentation to actions will be less meaningful to a user than a device which presents independent items, in this case, objects.

It seems plausible that some of these properties might play a role in determining the usability of an interface, particularly with respect to the initial presentation of entities and required syntax.
4. Implications of the Two Views for the User Model

As there seems to be some evidence for believing that objects may have inherent properties which give them an advantage over actions (when presented to the user first and when occupying the initial position of the command syntax), both views should be addressed. In this section, the implications of the two views for the User Model will be described.

Both views concern the processes at the Task Level which determine the ordering of the user's representation of task goals.

Under the Entity View, which asserts that an object => action syntax and an action => object syntax are equivalent in their ease-of-use, there would either be no ordering constraint present at the Task Level or the ordering process would be governed by factors such as task structure. If present, it would be equally likely that the ordering process would specify the sequence, object => action, as it would the sequence, action => object.

Under the Object Advantage View, which asserts that the properties of objects determine that they should occupy the primary position in a command sequence, the ordering process would specify that the task goals should be arranged in the sequence, object => action.

It is thought that an object advantage might not be observed under conditions where the requirement for subjects to access knowledge about object and action concepts is minimal. In the previous empirical studies, it has been noted that subjects can perform the tasks without having to generate the identity of the elements of the task goals from the knowledge that some goal state must be achieved. It is proposed that in a real world work situation, a worker would be required to undertake this form of processing. For example, a university researcher may receive a phone call asking for advice about recent references in his area. To accomplish this task, he might have to deduce that (at the very least) he must locate a list of recent references and arrange for it to be sent to the address of the caller. The level and amount of processing undergone in this instance is obviously much higher than if the request had been couched in terms such as, '(send to x) (list of recent references)'.

5. Future Directions

The ideas which have been discussed in this chapter suggest a number of directions for further work, some of these are described below.

i) to investigate, both empirically and analytically, the properties of objects which might
determine that they enhance ease-of use. This suggestion outlines a broad area of work concerned both with psychological research in the area of knowledge representation and relating the results of such work to the usability characteristics of devices.

ii) to provide an empirical test of the appropriateness of the two views expressed in the chapter. This direction has a number of problems associated with it concerning methods of isolating and manipulating variables which are internal to the user (such as topic assignment).

iii) to expand on the experimental studies reported to date, by varying the conditions to attempt to elicit some evidence for an object advantage. It has been suggested above that advantages specific to objects will not be observed unless users are required to process (deeply) the meaning of particular object and action concepts, then any processes relying on interdependencies etc. will be activated. This method of study could not provide any firm evidence concerning the appropriateness of the two views, however, it does provide an opportunity to see whether any object priority effects are active. If no object advantage is observed, there will be more reason to judge that the Object Advantage View is misguided (at least with respect to syntax).
Chapter 9
The Effects of Non-specific Instructions on the Usability of Alternative Syntax Structures: the Third Experimental Study (E3)

1. Introduction

In this chapter, a third empirical study is reported. The usability of two alternative syntax structures is explored under conditions which approach those found in a real world office.

Previous empirical studies have failed to demonstrate any advantage of an object => action syntax and imply that the usability of object-oriented interfaces is not a function of syntax. However, it is considered the results of these studies could be related to their laboratory style conditions.

In this study an attempt was made to simulate office conditions by:

i) requiring subjects to perform a number of non-computer supported tasks - these were included so that the subject's attention could not be focussed exclusively on the performance of the computer tasks.

ii) instructing subjects verbally, over the telephone, with task instructions which were worded very generally, i.e. the objects or actions which were to be used were not named. It was intended that subjects find it necessary to make inferences to determine what to do (i.e. the task goals).

Performance was compared on two versions of a simulated interface which differed with respect to command syntax required. The design and method of the study are described. An experimental hypothesis is expressed in relation to the two views proposed in Chapter 8.

The results of the study are analysed and discussed in terms of the two views. The results are then used to modify the User Model. It is found that the User Model resembles that expressed as the Rudimentary User Model described in Chapter 5.

The study concludes that there is some evidence for a superior performance when an object => action syntax is used and hence object-oriented interfaces would be considered more usable, at least in the domain of common office tasks.
2. Aims of the Study

The study aims to explore whether under 'naturalistic' conditions, an object first and an action first are equivalent in terms of user performance, or whether the object first syntax is advantaged in that it facilitates performance.

3. Background

In the previous chapter, Chapter 8, the discrepancy between results of the studies concerning syntax was identified and discussed. Two views were proposed which might account for the results. The difference between the views concerns whether objects have inherent properties which are facilitative for user performance when an object first syntax is required. A requirement to explore this issue was expressed.

It has been noted that in the two previous empirical studies (Chapters 6, 7) the experimental conditions were low in external face validity. For the sake of control, both of these studies tested user performance on artificial interfaces of limited functionality, under laboratory conditions. The subject had to perform a set of relatively straightforward tasks, one after another, in which all his or her attention could be focussed on the performance of the task. These conditions do not represent the way in which the tasks might be performed in the workplace and they differ from the conditions of the observational study. It could be argued that this difference in conditions is responsible for the discrepancy in results between the empirical studies and the observational study.

In particular, it may be the case that subjects working under the conditions of the empirical studies are not required to implement the full set of processes and representations which normally operate in the performance of such tasks. Instead, they could still succeed by employing a sub-set of processes. In the case of the first experimental study, subjects could employ a strategy of pattern matching, i.e. selecting device options to match those of the instructions; which would not require them to comprehend the full semantics of the objects and actions involved. For example, if they were told to send a memo, they would simply search for the entities 'Memo' and 'Send'; they would be unlikely to think about what a memo is, what it means to send something etc. It could be argued that these subjects were not performing in an appropriate context. Their context was one of a laboratory 'matching' test, rather than one of an office simulation. In terms of the User Model, these subjects would not be expected to apply a process of topic assignment at the Task Level and so their Task Goal representation would be unordered. Thus, the use of either syntax would not be prejudiced by
this representation. For the subjects in the second empirical study (in which task structure was varied), the User Model showed the operation of a Task Level ordering process, but the order determined by this process was entirely governed by the task structure and so it can be assumed that subjects were working in accordance with the given task structure, rather than inferring the task goals for themselves.

A third study will be designed to test the usability of the two syntax structures (object => action; action => object) under conditions which are more representative of an office simulation. An attempt will be made to immerse subjects in an office situation by altering the environmental conditions and requiring the subjects to perform office-based tasks additional to those under test. The form of the instructions for the tasks will be carefully designed to avoid the possibility of success through pattern matching, or use of other inappropriate strategies. It was expected that, due to the requirement for inference making and other reasoning activities concerning the task elements, subjects would apply a Task Level process of topic assignment. The process would determine the ordering of the Task Goal representation. Control of device features is still thought to be important and so a simulated interface will be developed. However, the structure of the interface will aim to reflect the semantics of the entities involved. Legal object-action pairings will mirror the pairings found in the domain of application.

Details of the simulated interface and experimental method are given in the following sections.

4. Details of the Simulated Interface

As in the previous studies (Chapters 6 & 7), the simulated interface was developed in HyperCard. Two versions of the interface were generated. These differed with respect to their syntax (object => action or action => object) and as a consequence, in the items appearing on an initial screen (objects or actions).

The interface was based on a pop-up menu structure, so the initial screen contained only eight entities (either objects or actions), each of which was associated with a pop-up menu.

The eight objects were:

- memo
- folder
- letter
- card-index
- directory
- graph
- diary
- account sheet

243
The eight actions were:

<table>
<thead>
<tr>
<th>print</th>
<th>copy</th>
<th>send</th>
<th>edit</th>
</tr>
</thead>
<tbody>
<tr>
<td>view</td>
<td>file</td>
<td>erase</td>
<td>sort</td>
</tr>
</tbody>
</table>

The number of items in a pop-up was not the same for each entity, but the two interface versions had pop-ups of equivalent size. In the Action version of the interface, the 'Print' pop-up had six items (letter; graph; diary; account sheet; memo; directory), while the 'Sort' pop-up had three items (folder; card-index; directory). The corresponding pop-ups for the Object version of the interface were those for 'Memo' (copy; edit; print; view; send; erase) and 'Diary' (view; edit; print). This arrangement was chosen to give the interface a 'natural' feel, i.e. that the action-object pairs had not been decided arbitrarily, but that they reflected the semantics of the particular items.

The initial screen of the interface is shown in Figure 9.1.

![Initial screen of interface for Action first group](image)

In each version, it was possible to perform the same functions, where a function is thought of as an object-action pair, e.g. print-memo; sort-folder. To perform a function, the subject had to select one of the items on the screen, causing the pop-up menu to appear, then move to select one of the items from the pop-up menu. The pop-up menus operated in a standard
way, i.e. they only appeared as long as the mouse button was depressed, letting go of the mouse button while one of the items on the menu was selected (highlighted) caused that item to be executed etc. Once a complete function had been selected, a new card was displayed which depicted the selected function e.g. a card displaying a letter and a print dialogue box.

As with previous interface simulations, the cards which represented the functionality of the system were identical for both variants. Also, both variants had some data capture facilities. The name and time of any screen item selections were recorded and the name and time for menu items were recorded. Subjects' choices were recorded whether they executed a complete function (e.g. send-letter) or whether they just accessed a pop-up (e.g. the 'Send' pop-up).

5. Method

5.1 Design

A between subjects design was chosen. The independent variable for the study was command syntax (object => action; action => object). Subjects were assigned to one of the two syntax groups.

The dependent variables were:

i) time taken to complete tasks
ii) number of correct tasks
iii) number of commands used (the number was compared with the minimum possible number of commands which was 16)
iv) number of times subjects accessed pop-up menus
v) subjects' comments

The experimental hypothesis for the study is couched in terms of the two views proposed in Chapter 8. While the study is not intended to provide a direct test of the views, it is expected to shed light on their appropriateness.

Under the Entity View, no difference in performance between the two syntax groups would be expected. Conditions which have previously been found to influence the usability of the alternative syntax structures (i.e. input style; task structure) are equivalent and neutral with respect to syntax in this study. In terms of the User Model, there would be no
interference from knowledge of English for either group, thus their representations of the commands would be equivalent, i.e. neither group would have to perform an additional 'Re-order' process to counter negative transfer. In addition, as the task structure is neutral, no 'Ordering' process would be expected to operate on the Task Goal representation. Therefore, the two groups would be expected to have equivalent representations.

Under the Object Advantage View, the Object first group would be expected to perform better on the measures taken because objects have properties which enhance their probability of being recruited as a topic (e.g. objects have a greater propensity for hiding; objects are more independent in their meaning etc.). Therefore, a requirement for an object => action syntax, promotes the direct correspondence of the representations of the task held by the user and the device. In terms of the User Model, subjects would be expected to apply an 'Ordering' process at the Task Level which would determine that the object would appear first in the Task Goal representation. For subjects in the object => action syntax condition, the ordering of this representation would be consistent with the ordering required in the representation of the commands used, therefore, no re-ordering would be necessary. Contrastingly, subjects of the action => object syntax group would have to apply an additional 'Re-order' process to arrive at a suitable representation of commands, thus their performance would be relatively impaired.

5.2 Subjects

Sixteen subjects took part in the study. They were all students or research workers from University College London. Participation in the study was voluntary and subjects were paid £2.50 for a session. The mean age of the subjects was 28 years with a range of 21 to 41 years. Four subjects were female and twelve were male.

All subjects had used a computer before with the length of their experience ranging from six months to twelve years. All of the subjects had used a computer for word-processing and some had used other applications and games packages.

Subjects were assigned randomly to groups.

5.3 Environment

The study took place in two rooms. The subjects were located in an office room with a window, desk, filing cabinet, computer and telephone. Subjects were given access to
materials which they might need for the tasks: pens, paper, labels, paperclips etc. Subjects were encouraged to organise themselves as they wished and to use anything they needed. There was a video camera in the room which recorded a general view of the subject and his or her work space.

For the test session, the experimenter used the other room (also an office). In this room, the experimenter could watch the subject's performance on a video monitor. The experimenter used the telephone in this room to communicate with the subject and to give instructions for the computer supported tasks.

5.4 Tasks

All the tasks were drawn from the domain of common office tasks. There were two distinct types of task; non-computer supported tasks and computer supported tasks.

The non-computer supported tasks included filling out expenses claim forms; writing address labels and sorting a card index. They were included for two reasons:

i) to provide contextualization; i.e. to give subjects a fuller impression of working in an office;

ii) to distract the subjects' full attention from the performance of the computer tasks. In previous studies, subjects have found performing office tasks in a laboratory situation easy, because they can direct their full attention towards the interaction. This situation is less likely to occur in a real-world office environment, where interruptions and competing demands are at work. The use of on-going non-computer tasks was intended to approximate real office conditions.

Subjects were given a 'To Do' list which described the non-computer tasks. They were asked to work through the list at their own pace.

The instructions for the computer supported tasks were given over the telephone. Subjects were informed that they would not have to use the computer except when they received a telephone call. The first task was presented three minutes after the beginning of the test session and subsequent tasks were given at five minute intervals.

The task instructions were given verbally to reduce the bias of a written syntax. It was hoped that subjects would assimilate an idea of what they had to do from the instructions in a natural manner. The instructions were expressed in general terms i.e. they were suggestive of what the subject had to do without being explicit. The actual object and action
implied were never mentioned. A number of pilot trials were necessary to compile an
appropriate set of instructions. While some of the tasks were more readily understood than
others, no task was performed incorrectly by all the subjects. Therefore, each task was
considered possible.

There were eight computer supported tasks each requiring the pairing of one object with one
action. There was no necessity for subjects to use any object or action more than once. This
arrangement was chosen in order to avoid a topicalisation effect. For example, if subjects
were required to perform a number of actions on the same object then a bias in favour of object
=> action syntax would be expected (see Chapter 7).

Task instructions are given in Appendix K.

5.5 Training

Subjects were given a general training in order to familiarize them with the interaction
techniques which they would use in the test session e.g. using the mouse; scrolling; selecting
from menus etc. Subjects were encouraged to practise until they felt confident about each
technique. The training materials were generated in HyperCard.

5.6 Experimental Session

The experimental session was the same for all subjects and it followed this schedule:

I  Questionnaire & General Instructions
II  Training (10 minutes)
III Scenario description and Test Session Instructions
IV  Test Session (40 minutes)
V   Debriefing

The entire session lasted for approximately one hour.
See Appendix L for the scenario description and the instructions given to subjects.

6. Results

A number of measures of subject performance were recorded. It was of interest to find out
whether the performance of the two groups differed on these measures and if so, in what
way. For each measure, the data were analysed using a two-way repeated measures analysis of variance with one between subjects factor [Group] and one within subjects factor [Task]. The results of these tests are described in the following sections; two global measures are reported first (number of correct responses on first attempt; time), followed by two measures, more specific to the nature of subject performance (number of commands selected; number of pop-ups accessed).

6.1 Number Correct at First Attempt

The number of tasks performed correctly was not analysed because most of the subjects did succeed on each task, however there were differences in the number of attempts which they made to achieve it. Instead the number of correct command sequences (consisting of an object-action pair) which were selected at the first attempt, was scored for each subject. A score of 1 was assigned when the subject selected the correct command pair on the first attempt and a score of 0 was assigned when he or she did not succeed at the first attempt. Throughout the scoring, the accessing of single pop-up menus, without selection, was disregarded and an initial 'View - OBJECT' pair was permitted. Most subjects chose to view an object before they performed any other action on it, and this tendency was assumed to be a type of 'checking' behaviour.

The mean number of correct sequences selected at first attempt by each group is shown in Table 9.1.
Table 9.1
Mean number of commands correct on first attempt per task, for each group

<table>
<thead>
<tr>
<th>Task</th>
<th>Object first</th>
<th>Action first</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.375</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>0.875</td>
<td>0.25</td>
</tr>
<tr>
<td>3</td>
<td>0.875</td>
<td>0.875</td>
</tr>
<tr>
<td>4</td>
<td>0.5</td>
<td>0.375</td>
</tr>
<tr>
<td>5</td>
<td>0.75</td>
<td>0.5</td>
</tr>
<tr>
<td>6</td>
<td>0.5</td>
<td>0.25</td>
</tr>
<tr>
<td>7</td>
<td>0.125</td>
<td>0.625</td>
</tr>
<tr>
<td>8</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>TOTAL</td>
<td>4.75</td>
<td>4.125</td>
</tr>
</tbody>
</table>

The analysis of variance revealed that there was no significant difference between the groups on this measure, \(F(1,14) = 0.64, p>0.05\) and that the interaction (task x group) was also not significant, \(F(7,98) = 2.06, p>0.05\). However, there was a significant difference between tasks, \(F(7,98) = 2.62, p<0.05\).

These findings suggest that some tasks were more readily understood than others, i.e. that correct command sequences could be selected straight away for some tasks, but other tasks were rarely completed on the first attempt. This effect seems to have been the same for both groups as there was no significant interaction between the two factors, task and group.

6.2 Time

The time taken (in seconds) by each subject to complete each task was recorded. The time for a task was taken to begin when the subject replaced the 'phone receiver, after hearing the instructions, and end when the subject left the computer and resumed work on the on-going non-computer task.

The mean times per task for each group are shown in Table 9.2.
Table 9.2
Mean time taken (in seconds) for each task, by group

<table>
<thead>
<tr>
<th>Group Task</th>
<th>Object first</th>
<th>Action first</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>69.5</td>
<td>68.25</td>
</tr>
<tr>
<td>2</td>
<td>79.875</td>
<td>159.375</td>
</tr>
<tr>
<td>3</td>
<td>99.375</td>
<td>85</td>
</tr>
<tr>
<td>4</td>
<td>88.5</td>
<td>100.75</td>
</tr>
<tr>
<td>5</td>
<td>79.75</td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td>131.5</td>
<td>149.375</td>
</tr>
<tr>
<td>7</td>
<td>141.375</td>
<td>98.625</td>
</tr>
<tr>
<td>8</td>
<td>83.125</td>
<td>112.375</td>
</tr>
<tr>
<td>TOTAL</td>
<td>773</td>
<td>878.75</td>
</tr>
</tbody>
</table>

The analysis of variance showed no significant effects, (p>0.05), for any of the factors, [Group: F (1,14) = 0.74; Task: F (7,98) = 1.91; Interaction: F (7,98) = 1.13].

6.3 Number of Commands

For each subject, the number of commands (object words or action words) which they selected in order to perform each task was recorded. It was expected that this measure would be informative about the relative efficiency of performance of the two syntax groups. For the most efficient performance possible, sixteen commands should be used - two per task, one object and one action.

The mean number of commands used by each group, for each task is shown in Table 9.3.
Table 9.3
Mean number of commands used for each task, by group

The analysis of variance showed there to be a significant difference between groups, [F (1,14) = 6.87, p<0.05]; but no significant differences between tasks [F(7,98) = 1.46, p>0.05] or the interaction [F (7,98) = 0.93, p>0.05].

On this measure, the Object first syntax group were found to use significantly fewer commands to complete tasks, than the Action first syntax group. From this result, it was inferred that the Object first group were more efficient in their performance of the tasks.

6.4 Number of Pop-ups

In addition to the total number of commands used, the number of times that a subject accessed a pop-up, without selecting a command from it, was recorded. Subjects were found to access pop-ups when they were 'looking' for an item. This behaviour suggests that they did not have a well formed view of which pair of commands they wanted, prior to interaction.

The mean number of pop-ups accessed per task for each of the groups is shown in Table 9.4.
<table>
<thead>
<tr>
<th>Group Task</th>
<th>Object first</th>
<th>Action first</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.25</td>
<td>2.375</td>
</tr>
<tr>
<td>2</td>
<td>0.875</td>
<td>3.125</td>
</tr>
<tr>
<td>3</td>
<td>0.75</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>0.625</td>
<td>1.625</td>
</tr>
<tr>
<td>5</td>
<td>0.625</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>0.75</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>0.75</td>
<td>3.125</td>
</tr>
<tr>
<td>8</td>
<td>0.625</td>
<td>4.75</td>
</tr>
<tr>
<td>TOTAL</td>
<td>6.25</td>
<td>23</td>
</tr>
</tbody>
</table>

Table 9.4
Mean number of pop-ups accessed on each task, by group

On this measure, the analysis of variance showed there to be a highly significant
difference between groups \([F (1,14) = 11.07, p<0.01]\), but no significant effects of task or the
interaction between task and group, \([\text{Task}: F(7,98) = 0.45, p>0.05; \text{Interaction}: F (7,98) = 0.54, p>0.05]\).

The results show that the Object first group accessed significantly fewer pop-ups during
their performance of the tasks, than the Action first group. The finding suggests that the
performance of the Action first group was characterized by a greater degree of searching
and uncertainty about the relationships between items. It appears that the Action first
group had a less efficient strategy for selecting appropriate command pairs for the task.

6.5 Subjects' Comments

After, the test session, the subjects were encouraged to give their opinions about the
usability of the device to which they had been assigned.

The subjects in the Object first group, in general, gave positive feedback concerning the
device. Subjects 4, 6, 7, 11 and 13 reported that selecting the object first posed no problems
and that they found the meaning of the commands and the mapping from instructions to
task quite obvious. Subject 4 commented that the objects had a familiar label, while the
actions seemed to be more 'computerish'. Subject 13 added that the objects were relatively
concrete as compared with the actions which were more ambiguous.

The majority of subjects in the Action first group, indicated that they found the device difficult to use and they were often confused. Subjects 3, 5, 8, 9, 12, 14 and 15 all reported that they wanted to find the object first and that they used the 'View' pop-up to find out what the objects were. Subject 9 said that he'd rather have objects immediately available, because, "it's the way you think, when you're asked to do something, there's one thing (object) and it has a label - there are more ways of expressing actions, they're more ambiguous". Subject 8 reported not being able to guess the action (from the instructions). Subject 5 said that, "the 'thing' stuck in my mind in the first instance", and he reported looking in the 'View' pop-up to identify the object and then looking for the action second. Subject 12 said that he didn't realise that the actions had an object set. In addition, he felt that the actions were non-unique for example, 'file' can mean a number of things.

7. Discussion

The findings of the study will be summarized briefly. There were no significant differences between the performance of the two syntax groups on the overall measures; number of correct command sequences selected at first attempt and time to complete task. The measure of the number of correct command sequences at first attempt, did reveal a significant difference between tasks, but this did not vary with group and it was assumed, therefore, that conditions were equivalent across groups. On the measures more specific to the nature of task performance, significant differences were found between the two groups. The Action first group used significantly more commands and accessed more pop-ups (a possible index of searching behaviour) than the Object first group. Comments made by the subjects concerning the use of the devices were consistent with the inferences made on the basis of the results of the analyses.

7.1 Implications of the Results: Which View is Supported?

The aim of this study was not to provide a direct test between the two views described in Chapter 8. Instead, the study aimed to compare the usability of two syntax structures under conditions which were more naturalistic, i.e. where subjects were encouraged to act in the context of an office situation (rather than a laboratory one) and where task instructions were expressed at a non-specific level, thus encouraging the proper formulation of task goals. It was hoped that the comparison would be informative with respect to the relative
appropriateness of the two views to the usability of object-oriented interfaces.

The experimental hypothesis asserted that under the entity view no difference in performance would be expected as conditions expected to influence syntax were equal and neutral (task structure; input technique), while under the object advantage view, the Object first group would perform better because objects (when prioritized in syntactic structures) have particular properties which facilitate the mapping between task goals and the computer representation of the task (these properties could be higher capability for hiding; a standalone meaning, i.e. they do not have to combined with other concepts to be understood; etc.). The results suggest that the Object first group did experience a facilitative effect as their performance was more efficient (on two measures) and they reported positively about the task demands. By contrast, the Action first group seem to have had difficulty in making the mapping between their task goal representation and the device representation of the task. In particular, they often searched for the correct object, before deciding upon the appropriate action. Their comments indicate that the object was the more salient entity derived from the instructions and that they wanted to specify this first, before selecting the more uncertain action.

On the surface, these results might seem to be more supportive of the Object Advantage View. However, the possibility of a difference in conditions enhances the plausibility of the Entity View. While the task structure, which has previously been shown influence the usability of particular syntax structures, was constant across the two groups, i.e. no entity was repeated within a task, possible effects due to other task level factors cannot be ruled out. It could still be the case that subjects 'topicalized' objects, in favour of actions. Under the Entity View, the usability of the two syntax structures would still be equivalent, and the observed difference would be explained in terms of asymmetric topic effects. However, it remains to be specified why objects should be prioritized by being assigned to the topic, when they feature no more frequently in the task instructions, than actions.

It appears that a view such as the Object Advantage View must also be recruited to explain why objects are more likely to be assigned to topic, in this situation.

There are a number of properties, inherent to objects, which may account for objects being more readily recruited as the topic of a task. These are discussed below:

i) Objects enable a greater degree of hiding.
Objects seem to be amenable to representations which are less computer-orientated than the representations for actions. Actions are often represented in a form which is compatible
with the functionality of the device. This representation is not necessarily compatible with the corresponding action found in the domain. For example, the action 'send' in the domain of office tasks means that an item A is removed from its initial location and transmitted to a new location; frequently the device representation of the action 'send' is such that the item remains in its original position as well as being transmitted. Often the representation of 'send' is therefore, a type of concatenated copy and send function. Representations of objects appear to have a more direct correspondence with the counterpart in the domain.

ii) The meaning of an object can be specified without reliance on actions.
It has been suggested that objects are more 'standalone' concepts than actions, i.e. they can be described without reference to actions. Indeed, objects are often specified in terms of their properties. It seems that actions are more dependent on objects for the specification of their meaning. The meaning of transitive actions is often specified with reference to an object e.g. the meaning of 'send' is usually described in terms of the effect of that action on an object, i.e. that it causes an item to be transferred from one location to another. However, little empirical support has been found for this argument. The subjects in this study did report that the objects were more concrete and that the actions seemed to be more ambiguous and dependent on a pairing with a particular object.

It could be that the difference between objects and actions described is a consequence of the level of description. A particular object, such as 'Dave's memo', exists at the sub-ordinate level, i.e. it is more specific than memos in general and than textual documents etc. An action such as, 'send', actually exists at the basic level, until it is applied in a particular instance. In English, there is no method for expressing actions at a more specific level (i.e. those of an instance), without referring to the object. For example, in 'I sent a letter', the 'send' differs from that in, 'I sent the boy home', but the difference cannot be specified without the object. If different verbs were available to reflect differences in implementation, then the reliance on objects would be rendered implicit. In the Navaho Indian language, there are a number of different words which mean 'pick up' and their usage is dependent upon the properties of the item to be picked up (long and thin versus round and plump etc.).

iii) Certain domains may be biased towards object differentiation.
It is plausible that certain domains of application may be organised such that their objects are particularly distinguishable and therefore, salient to the performer of the task (as compared with the actions). For example, compare wordprocessing, where all the objects
are of similar type (e.g. text documents), but the actions are more distinguishable (e.g. create/type in; move; copy; delete; save; underline etc.) with common office tasks (of the type sampled in this series of studies), where the object types are relatively distinct (e.g. menu; report; card-index; folder etc.) and a small set of actions can be used to manipulate all of them (e.g. copy; view; print; edit etc.).

Under the conditions of this study, a performance advantage was found for an object first syntax. This object advantage can be explained in terms of a topic effect, i.e. that objects were recruited to topic and the topic is expressed first in the user’s representation of the task, rendering the object first syntax more representationally compatible. If a topic effect is responsible for the apparent enhanced usability of an object => action syntax, then the two syntax structures can be considered equivalent. It would be expected that the action => object syntax would be more usable under conditions where the action is recruited to topic. However, if there are properties of objects which ensure their recruitment to topic, in favour of actions, then an Object Advantage View would also be appropriate. In this study, user comments supported the notion that there are properties particular to objects which are responsible for objects being recruited to topic.

7.2 Updating the User Model

The User Model which has been developed to reflect the results of previous empirical studies, will now be modified to take account of the findings of this study. The major difference between this version of the user model and the previous one (see Figure 7.4, Chapter 7), is that the formulation of task goals at the Task Level more nearly approximates the process as it occurs in an everyday context. The process is influenced by factors which are thought to be inherent to objects, rather than by factors which are experimentally manipulated (e.g. task structure). It has been suggested that objects have a predisposition towards being assigned as topics, at least in the domain addressed by this study. This predisposition might be related to the object’s ability to enhance hiding when it is placed first in a syntactic structure. The modified user structure is shown in Figure 9.2. The structure is similar to the rudimentary user structure expressed in Chapter 5 (Figure 5.1), as the ordering process in favour of objects has re-appeared. One difference in this structure is that it has to account for instructions given verbally. It is assumed that an equivalent perception process operates in the auditory medium to provide the user with a representation of single sounds (phonemes) which are later organised into words.
Figure 9.2

User structure for E3
In applying the structure to model instances of behaviour observed in the study, one difference is found between the two groups. The users interacting with the action-object syntax device are required to perform one additional process. At the Communications Level, they would be required to re-order their task goals, formulated with the object in the first position (i.e. object - action), in order to interact appropriately and specify the command sequence (action => object). It was inferred from the results of the study, that this additional phase was troublesome for many subjects. They appeared to act in accordance with their original task representation, i.e. they wanted to specify the object first and so searched through pop-ups in order to identify it, despite knowing that the action must be specified before the object. Subjects seemed unable to perform the necessary transformation prior to the initiation of the interaction.

The two user models, for the Object first group and the Action first group, are shown in Figures 9.3 and 9.4, respectively. The main features of the given devices used in the study, are that it accepts input in the form of mouse presses (i.e. an input style which does not influence syntax usability) and that the required syntax at the Communications Level is fixed and is object-oriented (object => action) in one instance and non-object-oriented (action => object) in the other.
Figure 9.3
Example of User model for Object first syntax group
Figure 9.4
Example of User model for Action first syntax group
8. Summary and Conclusions

Previous studies focussing on the usability of different syntax structures have yielded results which suggest that the two structures of interest are equivalent and that differences in performance can be attributed to the influence of factors such as input style and task structure. These results are not consistent with the results of the observational study where an object => action syntax was found to be preferred. The results also suggest that object => action syntax is not responsible for the ease of use of object-oriented interfaces. It is possible that the results of the empirical studies might be due, in part, to the experimental conditions employed. This chapter has reported a study in which the usability of two syntax structures has been compared, under conditions more similar to those found in an everyday office context.

Two groups of subjects participated in the study, one group was assigned to the object-first syntax device and the other to the action-first syntax device. The subjects were expected to perform a range of common office tasks, some of which were non-computer supported (e.g. sorting a card-index; form filling) and others, the test tasks, which had to be performed using the computer. Subjects were seated in a normal office environment with access to office stationery, computer and telephone. The subjects were instructed to perform the computer supported tasks by the experimenter over the telephone. Their performances over the eight computer supported tasks were compared in terms of number of correct commands at first attempt; time to complete task; number of commands used and number of pop-ups accessed.

There were no significant differences in performance on the overall measures (number of correct commands at first attempt; time to complete task), but there were significant differences between the groups on number of commands used per task (p<0.05) and number of pop-ups accessed (p<0.01). On both of these measures the Action-first group exhibited a less efficient performance. It was inferred that for these subjects, the task demands imposed by the device (i.e. specifying an action => object sequence) did not map directly onto the subjects' representation of the task. This inference was supported by subjects' comments. In the Action-first group, several subjects reported that they could not execute the task without 'locating' the relevant object first - even though, if they had located the correct action first, the object would be present in the pop-up menu. It appears that if the users' representation of the task does include ordering constraints (of the type, 'object specified first'), then it is difficult for subjects to overcome them and mentally re-express the entities in the order required by the device.

The findings were discussed with reference to the two views proposed in Chapter 8, the
Entity View and the Object Advantage View. It was found that neither view was supported directly, but that the results could be explained by combining the two views. Thus, each of the syntax structures is equally usable, given equivalence of conditions (e.g. task structure, pick & point style input), but that under conditions such as in this study, there is a tendency for the object to be recruited to topic and so an object first syntax is advantaged. It was not possible to decide from this study which of the properties of objects is responsible for their recruitment to topic. Comments from subjects were suggestive of the properties discussed in Chapter 8, that objects are more successful at hiding (they are less 'computerish') and that objects are more independent in meaning than actions (the meaning of a particular action appears to be specified with reference to particular objects etc.).

Thus, it appears there is some evidence that an object first syntax has advantages for task performance, at least in the domain of common office tasks.
Chapter 10
Overview

1. Introduction

This chapter provides an overview of the research reported in this thesis. The chapter attempts to assess the work with reference to the goals of the thesis, as stated in Chapter 1.

First the purpose of the chapter is outlined. The general approach taken in the thesis is then described.

The parts of the investigation relevant to each of the goals: characterization and assessment, are reviewed in separate sections. In addition, a number of general issues which have arisen during the investigation are discussed with respect to the thesis. The issues pertinent to characterization are:

- what is an object-oriented interface?
- can the device structure distinguish object-oriented interfaces from related types of interface?
- what is the relationship between object-oriented interfaces and object-oriented programming techniques?

The issues concerning assessment are:

- the ease-of use of object-oriented interfaces.
- recommendations for the design of usable interfaces.
- the usefulness of the concept, 'object-oriented interface'.

Following consideration of these issues, the limitations of the work reported with respect to each goal are presented and some possible future directions for the work are outlined.

Finally, the conclusions of the investigation are presented.

2. Purpose of Chapter

The purpose of this chapter is to review the work reported in the thesis with respect to its aims, as specified in Chapter 1. In Chapter 1, it was determined that the thesis would take the form of an investigation. The overall purpose of the investigation was to broaden the research base on
object-oriented interfaces from the perspective of cognitive ergonomics. The investigation had two goals:

i) to provide a characterization of object-oriented interfaces;
ii) to assess object-oriented interfaces with respect to their ease-of-use.

In the following sections, an overview of the work reported in the thesis towards these two goals will be provided. In addition, issues raised during the investigation will be discussed in the light of the findings.

3. The Approach

In Chapter 1, it was suggested that the work reported in the thesis would fall within the context of cognitive ergonomics. A cognitive ergonomics approach was outlined. In general, the assessment phase of the thesis did adhere to this approach. Throughout the empirical studies, the results were expressed in a user model which enabled the representation of the task held by the user and that held by the device to be compared. Ease of use was found to be enhanced when the two representations were compatible. A start was made towards identifying characteristics which influenced the compatibility between the representations, with respect to the feature, syntax. The results were intended to be informative for the future design of usable interfaces.

However, before the assessment phase could be undertaken, a view was needed of the nature of an object-oriented interface, a characterization phase was necessary. The thesis has two outputs, each associated with its goals:

i) provision of a characterization of object-oriented interfaces;
ii) empirical findings concerning ease-of-use of object-oriented interfaces.

These outputs are summarized below and a number of relevant issues discussed.

4. Characterization of Object-oriented Interfaces

At the outset of the investigation, it was found that while the term 'object-oriented' was applied frequently, there was no explicit definition and there was ambiguity concerning its application. Therefore, a need was established for work clarifying and specifying the nature of object-oriented interfaces. Analytic and empirical activities were undertaken towards the proposal of a characterization.
The following sections contain:
i) a brief overview of the output of the analytic and empirical activities;
ii) a consideration of the question, what is an object-oriented interface?
iii) a discussion of the adequacy of the characterization for distinguishing object-oriented interfaces from other classes of interface;
iv) a discussion of the relationship between object-oriented interfaces and object-oriented programming and development;
v) an outline of the limitations of the present work concerning characterization;
vi) suggestions for future work on characterization.

4.1 Overview of Output: Characterization

Analytic activity

The first step towards characterization was the proposal of a framework (Chapter 2). The framework comprised a basic set of distinctions and definitions and a structure to enable the modelling of devices and users. It was expected that the device structure would be used to determine the object-orientedness of any particular device. Thus, the device structure represents the major part of the work towards a characterization of object-oriented interfaces. The user structure was expected to be related to ease-of-use issues as it enables the comparison of representations of the task held by the device and the user.

The device structure enables modelling of devices with respect to dimensions critical to object-orientedness. It was expected that each dimension would represent a range of device features and that a critical feature value for object-orientedness would be specified for each dimension.

In Chapter 3, the initial content of the dimensions was determined by two methods:
i) an analysis of features of extant and prototype devices, which had been judged, intuitively, with respect to object-orientedness by domain experts (i.e. human computer interface specialists);
ii) a review of literature on related topics (e.g. object-oriented programming; object-oriented systems etc.) to enable abstraction of general features of object-orientedness.

These two strategies resulted in a tentative set of dimensions and critical object-oriented values. These are briefly described below.
Dimension 1: Representation of task entities
This dimension concerns the initial presentation of task entities to the user and the type of representation of the entities.
Object-oriented value: initial presentation of objects where objects contain information concerning both their identity and location.

Dimension 2: Limitation of command set
This dimension concerns the extent to which the command options available are limited to those that are legal or valid.
Object-oriented value: for each object, the command set is limited to those commands which can be performed on the object, given its type and current state.

Dimension 3: Syntax (of the command sequence)
This dimension concerns the syntax which is required by the device.
Object-oriented value: syntax is in the order, object => action.

Dimension 4: Object : action ratio
This dimension concerns the proportion of the number of object types represented by the device and the number of action types.
Object-oriented value: there is a greater object to action ratio.
However, it is possible that this feature is related more strongly to direct manipulation interfaces than object-oriented ones, see Section 3.4.

Dimension 5: Representation of object properties
This dimension concerns the way in which object properties are represented.
Object-oriented value: The properties of objects are represented such that they are immediately associable with the object (the contrast is to represent properties by the instantiation of functions).

Dimension 6: Concealment of mechanics of the programme
This dimension concerns the extent to which the user is shielded from the device specific procedures and is given the illusion that s/he is working with elements of the domain.
Object-oriented value: no device specific procedures are necessary.

In Chapter 8, the possibility of specifying object-oriented interfaces in terms of a superordinate feature, termed 'hiding', was considered. Hiding can incorporate the object-oriented values on a number of the above dimensions (e.g. dimensions 1, 2, 5, 6). Although potentially useful as a construct for specifying usability, no empirical validation of hiding has been undertaken.
It is proposed that any interface can be characterized with respect to its object-orientatedness through the application of the dimensions.

*Empirical activity*

The tentative device structure for classification was trialled empirically when it was used to distinguish two devices on the basis of their object-orientatedness. For the observational study, two extant interfaces were required which would support a common set of tasks, but which differed in their degree of object-orientation. The device structure was used to compare the relevant features of two devices, IBM TopView and Macintosh Lisa, on each dimension. It was found that Lisa was object-oriented with respect to five of the six dimensions. For example, on the third dimension, 'syntax', Lisa is found to have an object => action syntax. The dimension on which Lisa was not judged object-oriented was, 'representation of object properties'. Thus, overall, Lisa was characterized as being object-oriented.

As TopView (in standard use) could only be considered object-oriented in one respect, TopView has a limited command set, it was judged, overall, to be a non-object-oriented interface.

In addition, the observational study enabled a comparison of user performance on each of the two devices. Since the differences between the devices were well documented, inferences were made concerning the device features responsible for differences in user performance. In general, the evidence from this study supported the conclusion that the dimensions specified are not arbitrary with respect to user performance, but that they can predict particular effects on usability.

4.2 Consideration of the question, what is an object-oriented interface?

To evaluate the characterization activity, it is necessary to re-consider the question, 'what is an object-oriented interface?', in the light of the output summarized above. In the thesis, a dimensional approach towards a characterization has been taken and a device structure was proposed. While this device structure has been found useful in the comparison and judgment of interfaces as to their object-orientatedness, it is too detailed to provide a clear, overall view of an object-oriented interface. A definitional statement concerning the nature of an object-oriented interface is required. It is posited that the information necessary for a definition is implicit in the work of the thesis.
In an attempt to make explicit a definition, the two aspects of the term will be considered:

i) Object

ii) Oriented

Object

A preliminary definition for an object was proposed in Chapter 2:

An object can be described as the sum of the set of its features where features are the most primitive elements of the universe, i.e. they cannot be re-described in terms of more basic elements.

No restrictions were placed on the type of features involved to allow for the existence of different types of object e.g. abstract; concrete etc. Abstract objects might not have physical features. This definition of an object is very broad, but is it, alone, sufficient to capture the nature of the object implied in the context of object-oriented interfaces? In other words, when considering a dimension e.g. the representation of task entities, would this definition be sufficient for the identification of objects and would all objects fitting the definition have an equal status in the judgment of the interface. If not, some additional information must be specified by which the sub-set of objects can be determined.

On observation, it is found that three different object types can be identified in the computer interface.

i) There are objects which directly represent objects that are present in the domain of application e.g. a folder object in the office domain; an aeroplane in the Air Traffic Control domain etc. The method of representation is not critical - these objects can be represented either by icons or text.

ii) The computer also represents objects which are not present in the domain, but which sub-serve the computer version of the task e.g. a representation of the disk drive in the word-processing domain. The disk drive is not a necessary part of word-processing, which is concerned with the creation and modification of textual documents, but it is a means by which a storage facility is provided in the computer version of the task. In other versions of the task other mechanisms may be present, e.g. a filing cabinet.

iii) The third type of object found in computer interfaces is the display object. Certain components of the device display can be thought of as objects without recourse to their relationship with the domain e.g. icons; menus. An icon can be distinguished as an object in the display sense while its value in the functional sense might be either an object or an action. A screen object is simply a bounded area of the display which, when activated, has a reliable association with a particular
outcome. For example, a box containing the word 'PRINT' which is associated with the appearance of a Print Dialogue Box or alternatively, an icon depicting a document which is associated with a window displaying the text contents of the document. There is no stipulation as to the nature of the outcome (or function). It can represent the performance of an action, the presence of an object etc.

Note that the other two types of object are objects at two levels: at the display level and at the functional (or semantic) level. Objects at the functional level have properties which are amenable to change which is meaningful in terms of the domain, i.e. it represents the performance of a task. Objects, in this functional sense, contrast with actions which do not have properties but are the means by which the transformation of properties is effected.

Throughout the specification of the dimensions of the device structure, it has been implied that the objects required for an object-oriented interface contrast with actions, i.e. they are not simply objects in a display sense, but rather that they have functional characteristics of objects as well. For example, an interface which has an icon which represents 'Copy' would be judged to have a screen object (the icon), but not an object in the functional (object vs action) sense. If the interface represented the function using an icon of a copier, then it would be judged to have an object, in the functional sense. If screen objects were sufficient for object-orientedness, then WIMP interfaces would be classed as object-oriented as they, by definition, possess at least two types of screen object: icons and menus.

During the investigation, it has been suggested that screen display objects such as icons or text items which represent commands are thought of as entities, which might have a (functional) value object or action. Entities will be considered in more detail in connection with ease-of-use.

Oriented

In general usage, the term 'oriented' (synonymous with 'orientated') implies a degree of adjustment or alignment with some feature (e.g. circumstances or surroundings). In the case of object-oriented, 'oriented' indicates a degree of bias towards or focus on objects. In the context of interfaces there are a number of ways in which an interface might be oriented towards objects. Some of the ways in which an interface is oriented towards objects may be meaningful or interesting for the usability of the interface and others may not. For example, restricting the initial screen display to the presentation of objects would be expected to have important consequences for usability whereas displaying object labels in capital letters would not be expected to have such a distinct influence.

The dimensions of the device structure specify some of the ways in which interfaces might be
oriented towards objects. These types of orientation are thought to be important both for the classification of object-oriented interfaces and for the consequences which they have for ease of use. Thus, the dimensions can be used to assess the degree of orientation of an interface and the critical values aid the judgment as to whether the orientation is towards objects (as opposed to actions).

Definition
If the ideas concerning the two aspects of the term, 'object-oriented', are combined, then a definitional statement might be expressed as follows:

An object-oriented interface is one in which there is an overall bias towards presenting information in terms of objects. The bias may be manifest in a number of ways (as determined by the dimensions). An object is characterized as being the sum of the set of its features. In addition, objects are functionally distinct from actions, i.e. object properties can be transformed in order to perform a task within the domain, whereas actions are the means of transformation. Objects in this functional sense, can be distinguished from display objects or entities.

Contrasts
It is important to consider what sort of interface would provide a contrast to the object-oriented interface. In other words, how might an interface be considered non-object-oriented. To date any interface which did not pass the critical values on the dimensions for object-orientedness was classed as non-object-oriented. It was recognised that this class of non-object-oriented interfaces was not homogenous. Having reviewed notions concerning objects and orientedness, it can now be seen that there are at least two ways in which an interface can be non-object-oriented.

i) the interface can be oriented in terms of the dimensions, but the orientation can be towards entities other than objects, e.g. actions. Such an interface would be NON-OBJECT-oriented, i.e. action-oriented.

ii) the interface can be unorientated in terms of the dimensions. Here, there would be no special emphasis on either objects or actions, but instead the interface would have a mixture. Such an interface would be NON-object-ORIENTED.

4.3 Can the Device Structure Distinguish Object-oriented Interfaces from Related Types of Interface?

A requirement of the characterization, as stated in Chapter 1, was that it should enable object-oriented interfaces to be distinguished from other classes of interface. Each of the interface
types mentioned in Chapter 1 will be considered in respect of the device structure and the information concerning objects.

WIMP Interfaces

A WIMP interface which has windows, icons, menus and a pointer, is not oriented towards objects in the functional sense, though its elements suggest that it is oriented towards entities (icons). The value of the entities, object vs. action, is undetermined - either value can be present without altering the 'Wimpishness'. However, it might be the case that objects are more amenable to iconic expression than actions (in the functional sense), i.e. that it is more likely that icons will have the value, object, than action. (The problem of representing actions, statically, as icons was noted in Chapter 3). If icons tend towards having the value, object, then WIMPs must tend towards being object-oriented.

When the individual dimensions are considered it is found that a WIMP interface does not entail object-oriented features on any dimension. However, it can be seen that each of the elements of a WIMP interface can be supportive of an object-oriented feature.

For example, menus can support the object-oriented value, 'limited command set'. However, menus are not restricted to this function, they can be used to present a number of different types of limited option sets, e.g. a range of different spellings for a word; a list of different fonts etc. and neither are they necessary for presenting a limited option set. Windows can be useful to present views of objects, such that each object is uniquely associated with a particular window. Therefore, windows can be said to support the object-oriented value, 'task entities represented as objects', however, windows are not necessary for the instantiation of this value, nor are they restricted to this representation - windows can be used for many types of display, e.g. dialogue boxes can be thought of as a type of window.

The relationship between the dimensions and object-oriented values and WIMP interfaces is one of convenience. WIMP methods of presentation can often be used to support the instantiation of the values, but they are neither necessary for, nor restricted to, implementing object-oriented features. If WIMP interfaces are considered to be entity-oriented, then their ease-of-use might be attributed to their general support of recognition memory (as opposed to recall), rather than any specific features concerning objects or actions. It is easier for a user to select an item from the options presented than to recall it from memory, irrespective of whether the item is functionally an object or an action.
Metaphorical Interfaces

Metaphorical interfaces are primarily based on entities which are directly representative of those entities found in the domain of application. Therefore, whether their orientation is towards objects or actions depends upon the relative incidence and primacy of the two classes in the domain. Thus, a metaphorical interface can be object-oriented, if the domain is centered around objects, but there is no necessity for a metaphorical interface to be object-oriented.

When considering metaphorical interfaces alongside the dimensions and critical object-oriented values, the dimensions on which object-oriented values might be found are those concerned with the device's representation of objects, (e.g. 'the representation of task entities', 'the object:action ratio', 'the representation of object properties' and to some extent, 'the concealment of mechanics from the user'). To achieve a faithful metaphor, a metaphorical interface must have some representation of the objects of the domain and their properties. Thus, a metaphorical interface may be coincidentally object-oriented in some respects, but there is no reason for it to be object-oriented in respect of the following three dimensions: 'limitation of the command set'; 'syntax'; 'object:action ratio'.

Equally, it is not necessary for an object-oriented interface to be metaphoric because its objects can be computer specific and need not simulate those present in an existing domain. The primary support of a metaphorical interface lies in its reduction of computer specific procedures and the opportunity for the utilization of existing domain knowledge. Neither of these methods of enhancing ease of use has a particular association with objects.

Direct Manipulation Interfaces

In considering direct manipulation interfaces, there seem to be two views, a naive view and a sophisticated view.

Under the naive view, direct manipulation is seen as a superficial interface property which has consequences only at the level of input and output techniques. Here, direct manipulation refers to a method of interaction where the user can 'directly manipulate' entities represented on the screen using a pointer device (via a mouse; joystick etc.). From this viewpoint, object-oriented interfaces can be distinguished from direct manipulation interfaces. Their common features are restricted to a representation of objects on the screen and to some extent, the concealment of the mechanics of the program from the user. Direct manipulation entails representations which are manipulable on screen, these can be of objects, but there is no requirement for them to be expressed as objects other than at the display level, i.e. an entity (which could represent an object, an action, a property etc.). Also, a direct manipulation interface would tend to protect the user from the mechanics of a program by requiring input in the form of simple, physical actions rather than
command strings or function key sequences.

A more sophisticated view of direct manipulation includes consideration of the consequences of having direct manipulation of entities. Under this view, issues such as the presence of objects, their degree of analogy with real-world objects and the representation of consequences of actions become important. Consequently, there is a potential for direct manipulation interfaces to be considered orientated in terms of the dimensions, and furthermore, orientated towards objects when the critical values are applied. Below, the dimensions of the device structure will be considered to discover to what extent the two classes of interface can be distinguished.

The first dimension of the device structure concerns the representation of task entities. An object-oriented interface is expected to have an initial display of objects, which convey to the user information about their identity and location. It is clear that direct manipulation interfaces must have representations of objects at some level of description. In order that a user can manipulate directly, there must be some screen representation of the item s/he is manipulating. Therefore objects must be present, at least at the level of entities. The functional value of an entity may be an object e.g. a 'memo' object; a 'folder' object, or the value may be an action, e.g. a 'cut' action; a 'save' action, or a property, e.g. a 'New York font' property, a '2-point thickness' property. In the device structure, as noted above, there is no explicit statement determining the level at which objects should be present. However, a requirement has been expressed that objects should be present at the functional level (object as opposed to action). It was found in the examples of object-oriented interfaces observed (Chapter 3), that the objects which are initially presented to the user have the functional value, object. So, a possible difference between object-oriented and direct manipulation interfaces might lie in the level at which objects must exist.

However, it is often implied that direct manipulation interfaces also represent objects in more than a display sense. In his early characterization, Shneiderman (1982) wrote that direct manipulation interfaces have, "visual representation of the objects of interest". This statement implies that the objects have some correspondence with the objects found in the domain of application. Further affirmation comes from Owen (1987), who writes, "The response embodied in the basic Direct Manipulation paradigm . . . promotes the illusion that users are carrying out actions themselves in a visible world populated by more or less recognizable domain objects and tools". The visual display of objects may also entail that the representation of object properties is as specified for object-oriented interfaces. Indeed, it has been suggested that there is a distinct sub-set of direct manipulation interfaces which are object-oriented (Owen, 1986; Verplank 1988). It would be expected that this group of interfaces would achieve the critical values for object-orientedness, thus they would be judged to have both aspects 'oriented' and 'object'. It is
possible that the remainder of the class of direct manipulation interfaces may be oriented, but that there is no necessity for them to be oriented towards objects, in the functional sense. However, no evidence on this point has been found.

It has not been specified that direct manipulation interfaces must have an object => action syntax, but it is often observed as a feature of extant interfaces. An object => action syntax seems to be an artefact of the interaction style. If interaction occurs directly (without recourse to commands), then the object on which a manipulation is to be performed must be specified prior to the instantiation of the manipulation. Of course, a 'Select' action must take place, but it is a part of the interaction style, rather than a true component of the command syntax. 'Select' actions are present in many interaction styles e.g. in a Command Line interface, a 'select' action must take place prior to each keypress.

The degree to which a direct manipulation might foster the illusion of working with objects, is relevant to the sixth dimension of the device structure, i.e. the degree of concealment of the mechanics of the program from the user. To be successful at providing the illusion of working with real objects, a device must conceal device specific mechanisms from the user, as these will not reflect the domain. Hutchins, Hollan and Norman (1985, 1986), in their discussion of direct manipulation interfaces, propose that directness is associated with an approach towards invisibility of the computer to the user. They warn that directness cannot be directly equated with ease-of-use, because difficulties may be experienced by the user, though these will be a product of the domain rather than of interaction with the computer. It is likely that direct manipulation interfaces, which have a high degree of 'directness', will be object-oriented on this dimension.

The dimensions, limitation of the command set and object : action ratio, can be considered with respect to a particular property of direct manipulation interfaces. Some authors, (e.g. Hatfield, 1986), have proposed that direct manipulation interfaces are characterized by having a large number of objects all of which are manipulable by a small, common set of actions. If such an assertion is the case, then direct manipulation interfaces would fulfil the object-oriented criteria on both the two dimensions above. By inspection, it seems that direct manipulation interfaces tend towards this arrangement. Often, many of the functions which they support are 'object-ized', for example the function, 'to copy', may be represented by an object, 'the copier' - copies are made by moving the item to be copied to the copier. This arrangement automatically reduces the number of separate actions, e.g. 'copy' and 'print' could both be achieved via 'move', and increases the number of objects (and object types), e.g. 'copier' and 'printer' are included. Shneiderman (1983) holds that a direct manipulation user interface presents its user with a set of visual representations of objects on a display and a repertoire of generic manipulations that can be
performed on any of them. This comment stresses that the action set is common to all objects.

It has been suggested that the dimension concerning object:action ratios might be more pertinent to the description of direct manipulation interfaces. Object-oriented interfaces do not entail this 'object-izing' of function. Instead they rely on each object being associated with its own, limited action set. A further difference between direct manipulation and object-oriented interfaces might be that the limited command set for direct manipulation interfaces is common to all objects, whereas in object-oriented interfaces each object has its own particular action set.

The device structure is successful at distinguishing object-oriented interfaces from WIMP interfaces and metaphorical interfaces. Some of the requirements for object-oriented interfaces have relationships with the features of the other interface types e.g. the features of WIMPs are often useful for implementing the object-oriented values, but it has shown that such features are not essential for an object-oriented interface. As for direct manipulation interfaces, when a sophisticated view is taken the distinction is not so clear. However, application of the device structure does suggest that the two interface types have a different focus and consequently, there are subtle differences between them in respect of some of the dimensions. The device structure has potential as a means of considering and articulating the differences between direct manipulation and object-oriented interfaces.

4.4 The Relationship between Object-oriented Interfaces and Object-oriented Programming.

As noted in Chapter 3, the view is sometimes proposed that object-oriented interfaces are the result of the implementation of object-oriented programming techniques. However, it has been determined that object-oriented interfaces are, in fact, independent of the language in which they have been programmed. Both object-oriented interfaces can be programmed in non-object-oriented languages (e.g. declarative languages; procedural languages etc.) and object-oriented programming can be used to develop non-object-oriented interfaces. Nevertheless, the nature of the relationship between object-oriented interfaces and object-oriented programming should be considered.

First, there is a 'convenience' relationship between them, i.e. it is convenient to produce object-oriented interfaces using an object-oriented programming technique. Studies concerning the relationship between particular programming approaches and the problems to which they have been applied, show that there is often a class of problem to which a particular type of approach is best suited, though a solution utilizing a different approach is not precluded. For example, as noted in Chapter 3, FORTRAN was developed for problems concerning numerical calculations which can be expressed in ordinary algebraic terms whereas SNOBOL was developed for
problems involving string manipulations.

Second, the concepts are related by their focus on a common feature - the presence of the object. The nature of the object in object-oriented interfaces and object-oriented programming will be compared.

As stated above, the object in the context of the interface, can be defined as the sum of the set of its features. In addition, objects in object-oriented interfaces must have the functional attributes of objects i.e. they must have properties which are amenable to certain types of transformation which are meaningful in terms of the domain (the transformations enable the performance of a task). The limitation on the types of transformation possible is determined by the features or properties of the particular object type. Dynamic aspects of objects, i.e. changes that they undergo, are determined by the interactions of the user, i.e. the user applies actions. In object-oriented interfaces there is a tendency for at least some of the objects to represent objects which exist as part of the domain, however, it is not necessary for all objects to have a domain counterpart.

The object in object-oriented programming has been described by Tesler (1986) as follows:

"In an object-oriented system, every module is an object, that is, a data structure that contains the procedures that operate upon it."

Here, the object includes details of permissible manipulations and the manner of instantiation of these manipulations is limited by the object's type. Objects are typed according to their properties.

So, the notion of an object being composed of properties which are amenable to a limited set of manipulations is common to both object-oriented interfaces and object-oriented programming.

In programming, the dynamics involving objects are achieved through message passing between objects. Clearly, object-oriented interfaces differ in this respect, unless the user is conceived of as an object which initiates message passing.

Objects in programming might reflect objects which exist in the domain. It is often argued that object-oriented languages permit programmers to capitalize on their knowledge of the real world. Booch (1986) argues that in using an object-oriented approach, systems are structured around the objects that exist in the programmer's model of reality. However, there are no constraints which require the objects to reflect the real-world and so there is potential for the use of abstract objects.
It appears that the underlying ideas concerning objects in programming and the interface are similar - they are not found to be inconsistent. Yet, there is no necessity for a direct mapping between instantiations of programming objects and instantiations of interface objects in a particular device. For example, the programme might include an 'Editor' object which handles all editing functions, e.g. display of dialogue boxes; updating on screen etc., but this object need not have a counterpart in the interface. Instead, the user might be presented with a collection of menu items associated with editing.

4.5 Limitations of the Work Concerning Characterization.

While the work reported can be viewed as useful groundwork in the characterization of object-oriented interfaces, three broad classes of limitation are apparent:

i) The scheme may not be adequately specified to be usable by other researchers.

This limitation is important as it restricts the utility of the output of the thesis. A particular problem is that for certain dimensions, it is difficult to specify an interface's value along that dimension. For these dimensions, a subjective judgment has to be made. The dimensions which are particularly vulnerable to this criticism are, 'the representation of task entities', 'the degree of concealment of the mechanics of the program from the user' and 'the representation of object properties'. If these dimensions are believed important, then further work is necessary to determine the possible values and their relative object-orientedness. It might be more satisfactory if the device structure included some guidelines as to how judgments should be made in respect of these dimensions. For example, guidelines could exemplify potential ranges of features along these dimensions and advise on the degree of object-orientation which should be assigned for each feature. In the absence of a varied set of object-oriented interfaces this undertaking would be difficult to achieve.

ii) The correctness and completeness of the information contained in the structure is not guaranteed.

This limitation may also restrict the utility of the output in its present form.

The dimensions and associated critical values were determined by observation and by reviewing the literature. All the extant interfaces observed were alike in that they had a direct manipulation style, were WIMP interfaces etc. so the range of features sampled may not have been sufficiently varied to capture all the relevant features for object-orientedness. The prototype interfaces observed were still under development and so it is possible that in later versions they
might acquire important features which have been overlooked in this investigation.

While the object-oriented features specified were observed in the examples studied, it cannot be guaranteed that they are pertinent to the distinction of the object-oriented class of interface from other classes. Indeed, one particular dimension, object:action ratio, seems to be more pertinent to the distinction of direct manipulation interfaces, than object-oriented ones. According to some authors (Hatfield, 1986), the extreme case of a direct manipulation interface is an interface in which the number of objects is vast and the number of actions restricted to a small common set. While, this dimension clearly articulates a method of bias towards objects, it is not certain that it is well represented in common exemplars of object-oriented interfaces nor is it clear that it contributes towards ease of use, especially for the naive user. This dimension might be particularly useful for identifying instances of the subset, object-oriented, direct manipulation interfaces.

There has been no rigorous examination of the inter-dependencies or other relationships between the dimensions with respect to object-orientedness. Possible relationships have been mentioned, e.g. initial representation of objects of the screen is necessary for an object-> action command sequence, if the input method is a Pick & Point style. However, the extent to which possible relationships contribute to the judgment of an interface as object-oriented has not been explored. This aspect of the limitation is important, as a true picture of an object-oriented interface may rely on a particular configuration of interdependencies. In the absence of a varied set of extant interfaces which can be judged object-oriented, it may be necessary to model alternative interfaces which vary systematically with respect to the dimensions to discover possible contingencies concerning the co-existence of features.

iii) The validation of the dimensions with respect to consequences for user performance is not complete.

Some of the dimensions were not addressed in the observational study, i.e. representation of properties and limitation of the command set. Additionally, only the feature, syntax, has been subjected to controlled experimentation. For further validation of the dimension set, there is a need to determine the relevance of all the dimensions to ease of use.

4.6 Future Directions Concerning Characterization

The suggestions for future work concerning characterization evolve from the limitations listed above. In general, there is a need to firm up the contents of the dimensions and to ensure that they
are both correct and complete. Further, if the dimensions are to be used for characterization by naive third parties, their expression and clarity need to be improved. Directions which would serve these purposes include:

i) Improvement of the present device structure by determining the relative importance of each dimension; specifying their nature in more detail and providing guidelines as to how judgments should be made considering some of the less quantifiable dimensions. Consolidating the dimensions in this way should improve their usefulness to third parties. Additionally, the most appropriate form for the structure should be determined, e.g. a table; pro-forma checklists etc.

ii) Application of the current device structure to a greater range of interfaces. This operation should provide an opportunity for broadening the scope of the dimensions. It would be particularly useful to observe interfaces from a wider range of domains and interaction styles to determine the extent to which the dimensions tap the critical elements of object-orientedness as opposed to the presence of confounding features.

iii) To perform further empirical tests of the dimensions to determine their consequences for user performance, particularly concerning the dimensions which were inadequately tested in the observational study, i.e. 'limitation of the command set'; 'representation of object properties' and 'object:action ratios'. This activity would enable the extraction of design guidelines based on the relationship between the range of features on the dimension and usability.

5. Assessment of Object-oriented Interfaces with respect to their Ease-of-use

The second goal: to assess the ease-of-use of object-oriented interfaces, was achieved through both empirical methods and analytic methods. The investigation progressed iteratively. Data were collected from each empirical study and were expressed in the form of a user model. Analysis of the findings in terms of this model helped to guide the direction of the series of studies. Two types of empirical study were employed:

i) an observational study - to provide general information concerning the ease-of-use of object-oriented interfaces.

ii) experimental studies - a series of studies were designed to provide specific information concerning a particular object-oriented feature under controlled conditions.

In the following sections, a brief overview of these studies and their output is presented; a number of issues are discussed in the light of the work reported in the thesis, the limitations of the research concerning assessment are considered and finally, some suggestions for future work are
5.1 Overview of Output: Assessment

The Observational Study

The observational study was intended to compare the usability of two interfaces, one object-oriented interface and one non-object-oriented interface, over an equivalent set of tasks. Comparison of the errors and difficulties experienced by the subjects was expected to yield information about the device features which influence usability.

It was found that, overall, the object-oriented interface was easier to use than the non-object-oriented interface. Subjects using the object-oriented interface made fewer errors, fewer requests for help and they completed more tasks successfully. The particular problems experienced by the non-object-oriented group were: navigation around the system; use of computer specific procedures and achieving the mapping between the effects of their performance and the intended task goal. It was inferred that the following object-oriented device features were important for ease-of-use:
- representation of task entities as objects (as opposed to actions);
- concealment of the mechanics of the programme e.g. automatic invocation of application programmes. Interaction can occur directly with the objects, the user does not have to be aware of switching between applications.

Note that TopView can be considered a non-oriented interface (i.e. that it is mixed), rather than being a directly action-oriented interface.

In addition, due to a particular feature of the non-object-oriented interface (TopView), that it enabled the use of a non-standard syntax, object => action, it was found that users chose to use this syntax, even though its use had never been demonstrated. From this observation, it was suggested that an object first syntax is easier to use.

The Experimental Studies

A single dimension was selected as a focus for the experimental studies. The decision to concentrate on one dimension was made to achieve a higher degree of control of variables. The dimension, syntax of the command sequence (Communications Level), was chosen for two reasons:
i) an object first syntax had been found to have consequences for user performance;
ii) the object => action syntax is a well-established and easily observable feature of
object-oriented interfaces.

The relationship between syntax and other features of object-oriented interfaces will be discussed
in a later section.

Special purpose simulated interfaces were used in all the experimental studies to ensure that the
devices being compared differed only in respect of their syntax.

Experimental Study 1

The advantage of an object => action syntax, as found in the observational study, was observed
under particular input conditions, where subjects had to select items from the screen using a mouse
pointing device (Pick & Point style). It is important to determine whether the advantage was
general or particular to this input technique. The first experimental study aimed to compare the
usability of two syntax structures (object first; action first) under two input conditions (Pick &
Point; Command Line). Contrary to expectations, no advantage was found for an object first syntax
under either condition. In the Pick & Point condition, users of the two syntaxes performed
equivalently while in the Command Line condition, the object first group performed worse than
the action first group, making significantly more syntax errors.

The conditions of this study and the observational study were compared to reason about the
different findings. It was proposed that the lack of an object first advantage for the Pick & Point
condition in the experimental study, was attributable to the straightforward task demands.
Subjects could perform the task by applying a simple match procedure. In the observational
study, subjects had to reason about the nature of the objects and actions involved in the task. In
terms of the User Model, it was proposed that such reasoning activity might result in the
occurrence of an additional process at the task level, assigning a focus or topic for the task. This
additional process was expected to influence the representational compatibility of the
alternative syntax structures tested. It was proposed that in the observational study, the object
had been assigned to topic which had favourably prejudiced the use of an object => action syntax.
If no such process was applied in the experimental study, then the representations held by both
syntax groups can be expected to be equally compatible with the required syntax.

Additionally, it was proposed that under the Command Line conditions, subjects must draw on
their knowledge of English in more detail in order to type (e.g. knowledge of spelling rules;
knowledge of English grammar etc.). The object first group would be expected to suffer an
interference effect from their knowledge of English syntax for the imperative (i.e. object =>
action) because this ordering is directly opposed to the syntax required by the device. It is
suggested that it is the need for production, i.e. typing, which activates this effect. The subjects
in the Pick & Point condition would be expected to be able to rely on more generalised knowledge
of English for the selection of items.

**Experimental Study 2**

The second experimental study aimed to explore the proposition that topic assignment at the task
level influences the usability of particular syntax structures. It was thought that there was a
tendency for users to want to express the topic (or superordinate goal) of a task first in any
command sequence. As topic assignment is difficult to manipulate, an alternative variable, task
structure, was employed as an indirect index. A task structure can be said to be object based if a
common object is repeated throughout its sub-tasks (e.g. edit the memo/ copy the memo/ print the
memo etc.) and conversely, it is action-based if a common action is repeated (e.g. file the report/
file the letter/ file the graph etc.).

Subjects were assigned to two syntax groups (object first; action first) and were given a number of
tasks to perform which varied with respect to their task structure. Thus, for some tasks the
syntax required and task structure would be consistent (e.g. object first syntax + object-based task
structure) while on others the variables would be inconsistent (e.g. object first syntax +
action-based task).

Results showed that there was an interaction between the two variables on two measures (time to
select first command & combined error and sub-optimal response scores). Performance was found to
be better on consistent trials. This effect was equivalent for the two syntax groups. In other words,
irrespective of the actual syntax required by the device, if the first item of the command sequence
is consistent with the repeated item of the task structure, then performance is facilitated. Again
no evidence was found for an object first syntax advantage per se. The study suggests that the
usability of the syntax structure depends on the task structure. In terms of the User Model, the
subjects would apply an ordering process at the Task Level which would depend on the given task
structure - the repeated item would be placed first. Representational compatibility would then be
greater when the device syntax required the repeated item to be expressed first.

These results were considered with respect to previous results. It was believed that the tasks used
in the first experimental study, which consisted of a single object-action pair, were neither object
nor action based. Thus, no effect due to consistency between task structure and syntax would be
expected. It was proposed that, in the observational study, subjects had naturally assigned the
object to a topic role thus facilitating the use of object => action syntax. However, it is also possible that the object advantage of the observational study is due to the action of a variable which has not been addressed in the experimental studies.

Experimental Study 3

In the third experiment, an attempt was made to reinstate the object first advantage by employing more 'naturalistic' conditions. Again the performance of two syntax groups was compared (object first; action first), but here the subjects were expected to infer the actual task goals from vaguely expressed instructions. The task instructions were given verbally to maximise the likelihood that subjects would have to internalise them to determine what they had to do. In addition, subjects were given non-computer tasks to do to ensure that their attention was not wholly directed on the computer tasks throughout the session and to provide a more realistic office context. Both of the conditions previously found to influence syntax usability were neutral in this study (input style and task structure).

Under these conditions, it was found that the object first group did perform better, showing a more efficient performance in terms of the number of commands used and the number of pop-ups accessed. From these results, it was inferred that in performing common office tasks, subjects tend to recruit objects to a topic role. Thus, an object first syntax is more usable as it provides a better mapping between the subjects' representation of the task and the representation given by the device. For the action => object condition, the subjects' representation of the task would contain an additional step determining the re-ordering of task goals. A number of reasons were posited for why objects might fulfil this topic role more readily than actions. Although there is a lack of empirical evidence it appears that objects are more independent in meaning than actions and there is reason to believe that they promote a greater degree of hiding. Subjects comments were found to be consistent and supportive of these inferences.

5.2 Are Object-oriented Interfaces Easy to Use?

In the first chapter, a requirement was proposed to discover whether object-oriented interfaces were as easy to use as their reputation had suggested. The results of the empirical studies, which addressed the ease of use issue, will be considered.

First, the term 'ease of use' will be discussed. There are a number of ways that ease of use can be manifest at the interface, e.g. the ease with which someone who has never used the device before can operate it; the ease with which all the operations of a device can be learned etc. In the thesis, it has been assumed that ease of use is related to the cognitive compatibility between the
user and the device. Particularly, that the ease of use an interface will increase in proportion to
the compatibility between the representations of the task held by the user and the device. It is
assumed that if the initial representation of the task held by the user has to undergo few
transformations to render it in an appropriate form for interaction with the device, then that
device will be easy to use. If few intermediary transformations are required then the task will be
accomplished quickly and there will be less potential for error caused by the user acting in
accordance with an inappropriate, intermediate representation. Thus, the general measures
employed in the observational study, i.e. time, errors, difficulties, tasks completed etc., were
assumed to be metrics pertinent to the assessment of ease of use. In the later studies, the results
were expressed in terms of the User Structure and models of the representations of the task held
by the user were devised. It was hoped that the models would permit a more direct assessment of
cognitive compatibility. The studies and their contribution to the assessment of ease of use will be
discussed below.

In the observational study, it was found that, in terms of the measures employed, the
object-oriented interface as exemplified by the Apple Lisa was found to be easier to use than the
non-object-oriented interface, TopView. A number of suggestions were put forward as to the
features of the Lisa which were responsible for its enhanced usability.

For example:
- representation of the task entities as objects;
- concealment of the mechanics of the programme;
- object => action syntax.

However, the results could be a product of the particular exemplars tested. There was no attempt
to expand the contrast by observing performance over additional interfaces varying in their
object-orientedness. Although, it was proposed that the performance differences were related to
the differences between the devices with respect to their object-orientedness (as specified using
the device structure), there is no guarantee that the differences were not due to the presence of
other factors. Since extant interfaces were employed, the object-oriented exemplar supported a
direct manipulation interaction style, so there is a possibility that the effects of object-oriented
variables were confounded. To eradicate this possibility it would be necessary to run a similar
study using a device which was as object-oriented as the Lisa, but which did not support direct
manipulation. Additionally, there is a possibility that the performance differences were due to
factors not thought important for object-orientedness or other styles, such as cursor shape; response
speed of screen display etc..

Despite these problems of experimental control, the results of the observational study remain
suggestive that object-oriented interfaces (as determined by the device structure) are easier to use than non-object-oriented interfaces. In particular, it was found that the object-oriented interface enabled straightforward navigation around the system, it helped subjects in knowing where they were and it reduced the number of meaningless (non-task-related) procedures which the subjects had to apply. In Chapter 8, a construct, hiding, was suggested which covered all these facets. It was proposed that orientation towards representation of objects enhanced the degree of hiding possessed by an interface. Although analytically reasonable, this proposition has not been tested empirically.

The experimental studies focussed on the usability of different variants of the single feature, syntax of the command sequence. While, strictly, the results of the studies can only be used to reason about the usability of different syntax sequences, they can provide insight about more general features of object-oriented interfaces. For a Pick & Point style interface to support an object => action syntax, representations of objects must be displayed on the screen. The initial display of objects on the screen is a further indicator of an interface's orientation towards objects. Thus, it is expected that results concerning the ease of use of an object => action syntax will be relevant to the consideration of the usability of object-oriented interfaces more generally.

Contrary to the results of the observational study, the results of the first two studies suggested that an object => action syntax was not more usable than an action => object syntax. In fact, the studies were supportive of the view that an entity-oriented interface was easy to use and that the functional value of the entities (either object or action) did not, per se, influence usability. Instead, factors which were found to influence usability were the type of input style and the consistency between the task structure and the required syntax. An entity-oriented interface would be expected to aid usability (as compared with a typed command style interface) because the user is only required to select from the entities displayed, thus, recognition memory is involved rather than recall. In the psychological literature it has been documented that recognition memory is better than recall (Luh, 1922). A number of factors have been proposed as responsible for the difference, e.g. recognition reduces the size of the set of alternatives from which the correct response must be drawn (Davis, Sutherland and Judd, 1961). Similar arguments have been proposed for the benefits of menu-driven interfaces over the command line interface.

However, in the third study, the advantage of an object orientation was reinstated. The results of the study suggested that the use of objects at the interface could enhance usability if the nature of the tasks and domain were such that objects were likely to be assigned to a topic role. The conditions of the study dictated that subjects reasoned about the objects and actions involved in the tasks and constructed an appropriate task representation. These processes, at least in the common office task domain, enable subjects to capitalize on the 'standaloneness' of object concepts.
and the commonality between the device’s representation of them and their own representation of objects from the domain (i.e. their ability to promote hiding).

Some initial thoughts concerning design guidelines can be drawn from the results of the assessment and these are expressed in the following section.

5.3 Recommendations for the Design of Usable Interfaces.

It was intended that information gained concerning ease-of-use could be used to inform design of future computer systems. Some implications for design derived from the work presented in the thesis are discussed below.

i) An orientation towards objects does not always enhance the usability of a computer system. Results of the first and second experimental studies strongly suggest that providing an orientation towards objects, with respect to syntax, does not increase usability under all conditions. This finding suggests that caution should be exercised in the orientation of a system towards objects - particular parameters should be taken into account. Some of these parameters have been expressed in the Summary Tables of Results for the experimental studies (Chapters 6 & 7).

For example, in the first experimental study, it was found that an object first syntax was inappropriate where the method of interaction was a command line style. For command line interfaces, the use of an action => object syntax is advocated as this maximises the compatibility of the user’s knowledge and the representations required by the device. In the second study, it was found that the two syntax structures were of equivalent usability. Here, task structure was found to influence the usability of a particular syntax structure. This finding suggests that the tasks of the application domain are carefully analysed to determine whether they tend towards having an object-oriented or an action-oriented structure. The design of the command sequence should aim to maximise the consistency between the task structure type found in the domain and the required syntax for the command sequence.

ii) To maximise the effectiveness of the use of objects at the interface, the nature of the domain of application must be taken into consideration.

The third experimental study suggested that an object => action syntax would enhance usability if there was reason to believe that the objects of the domain of application were susceptible to recruitment as topic. Domains where objects were more likely to be recruited to topic than actions are expected to be characterized by:
- a large number of objects;
- objects which are meaningful with respect to the goals of the task;
- objects which are well-differentiated from one another.

For example, it is expected that the office task domain is more likely to have objects which are susceptible to topic assignment than wordprocessing. In the office task domain, there is a number of distinct object types (e.g. folder; memo; graph; card-index etc.) each of which are meaningful with respect to task goals. By contrast, in the wordprocessing domain, the objects are mostly of the same type (text document) and the actions might be judged to be more well differentiated (e.g. save; format; copy; proofread etc.) and meaningful for the execution of task goals. In the case of wordprocessing it is likely that actions will be focal. There is an argument for characterizing the domain prior to determining the orientation of the interface (towards objects vs actions vs mixed).

Thus, the contribution in terms of reliable guidelines for design is small, but it has outlined an approach towards the determination of such design guidelines concerning object-orientation.

5.4 The Usefulness of the Concept, 'Object-oriented Interface'

In this section, the usefulness of the concept of an object-oriented interface will be considered. A number of ways in which the concept can be thought of as useful can be identified from the work reported in the thesis.

i) When defined, in terms of dimensions for orientation and critical values for orientation towards objects, the characterization of an object-oriented interface successfully delineates a class of interface which is separable from the classes defined by the terms, 'WIMP interface' and 'metaphorical interface'. Though, a particular interface may exhibit the features of more than one class, it might be a WIMP interface and also an object-oriented interface, the features determining the classifications are independent. Therefore, it is possible to observe an object-oriented interface which is neither a WIMP nor a metaphorical interface, and hence the term, 'object-oriented interface', has utility as a descriptor.

ii) To some extent the concept, 'object-oriented', is useful for distinguishing direct manipulation interfaces from object-oriented interfaces. Although the two classes have not been distinguished conclusively, consideration of the way in which an interface might be object-oriented has yielded aspects of object-orientation which are independent of direct manipulation. It has been found that there are examples of interfaces which have some object orientation but which are not direct manipulation interfaces (though all interfaces of this type have been prototypes as opposed to
commercial products). Also, the converse is true, that while direct manipulation interfaces might be oriented towards, for example, entities, it is not necessary for them to be object-oriented, unless they are members of the sub-class, object-oriented direct manipulation interfaces.

iii) Some of the features which have been proposed as important for the characterization of an object-oriented interface, have been found to play a role in the ease of use of the interface. The distinction of the class of object-oriented interfaces is more than an exercise in taxonomy as the features which are identified have been found useful for guiding the design of usable interfaces.

iv) It has been determined that it is possible to determine aspects of object-orientation at the interface which are independent of the programming approach employed. Object-orientation at the interface is not dependent on an underlying programme written in an object-oriented language. It is possible for objects to exist for the programmer which are not perceived as objects by the user at the interface. Conversely, interfaces which exhibit features of object-orientation can be programmed in languages which are not classed as object-oriented. Thus, the concept of object-orientation at the interface is useful for separating different aspects of the device, i.e. the internal aspects, (the code) and the external aspects, (the interface to the end-user).

From a number of standpoints, the concept 'object-oriented interface', represents a configuration of features which are not captured by other terms and it, therefore, can be judged useful. Furthermore, the nature of those features appears to be important in the pursuit of determinants of an interface's ease of use.

5.5 Limitations of the Work Concerning Assessment

Three broad classes of limitation have been identified in the assessment of the ease-of-use of object-oriented interfaces.

i) In general, the results of the empirical studies lack breadth. The studies were designed to follow a particular theme and in an attempt to arrive at a coherent answer there were only minor variations between the conditions of each study. The thesis has only begun to address the assessment issue and a much greater range of studies is necessary to provide an overall picture of their ease-of-use. Some aspects of this limitation are discussed below.

The experimental studies had a limited scope because they only addressed the ease-of-use of one feature of object-oriented interfaces. Clearly, series of studies could be designed to investigate the other proposed features of object-oriented interfaces. To test the effects on usability of the other
features, a similar method could be employed, i.e. developing simulations of interfaces which vary with respect to the characteristics of interest. For some features, e.g. the limitation of the command set, the development of alternative simulations would be straightforward. However, for the features which are more difficult to contrast (as noted in Section 3.5), e.g. concealment of the mechanics of the programme from the user, more work on characterization would be a necessary precursor to simulation development.

Although, an investigation of each of the proposed features of object-orientedness would appear to complete the assessment of object-oriented interfaces, the possibility of combination effects, or interactions, cannot be discounted. While some empirical work should be directed towards the exploration of isolated features, an investigation of combinations of features is necessary before a complete picture of the ease of use of object-oriented interfaces can emerge.

ii) The results of the studies cannot be generalized. While the results of studies are suggestive of a number of general facts concerning ease-of-use, they cannot be used as proof of those facts. Again, the number of studies and the range of conditions is too small. This limitation suggests that the thesis has demonstrated the topic to be worthy of interest and further work, as it has not been able to provide general answers.

For example, in the observational study, where an attempt at a general comparison of the usability of object-oriented and non-object-oriented interfaces was made, only two extant systems were employed. While these devices varied systematically on the dimensions suggested for distinguishing object-oriented interfaces, they still must be considered as representatives of their classes. The results of the observational study, while being suggestive that object-oriented interfaces are easier to use, strictly can only be used to show that Lisa was easier to use than TopView over the particular tasks tested.

In addition, each of the sets of conditions employed in the experimental studies was tested only once. In a single series of experiments results cannot be generalized beyond the actual conditions of the experiment (e.g. the particular tasks; command words; style of screen display etc.) with any degree of confidence. While attempts are made to balance these factors among the subject groups, the entire group is limited to exposure to one overall set of conditions. Here, it cannot be proven that, for example, objects are recruited to topic, except when the objects are memos; reports; graphs; letters etc. For generality of results, replication studies are necessary which permit variation of the incidental variables.

All the tests concerning the ease-of-use of object-oriented interfaces have concentrated on the domain of common office tasks. While it is true that some extant object-oriented interfaces offer
support for such tasks, this domain is not the only area of their application. To provide a complete assessment of their properties with respect to ease-of-use, they must be tested within the context of other domains. It has been proposed that the findings which suggest that objects are more likely to be recruited to topic may be dependent on domain (Chapter 9). To validate this claim, a series of studies focussing on alternative domains is necessary. The domain of application should be assessed analytically to determine its structure in terms of object or action precedence; hypotheses concerning topic assignment can then be formulated and tested.

iii) The development of the User Model risks suffering circularity. The models of the user were derived from the results of the empirical studies. The processes were inferred from the experimental metrics (e.g. errors; relative time spent etc.), for example, the presence of a higher number of syntax errors was taken as an indication that subjects were failing to undergo a necessary transformation and were responding in accordance with an intermediate transformation. While the inference is consistent with the results, without the benefit of direct access to cognitive processes, there is no proof that it is the case. There is a degree of circularity in the model development because the results are used to generate the models and the models are used to ‘explain’ the results. However, this limitation is not necessarily damaging because the models can be used predictively to determine expected results under a given range of conditions. Further work of this sort is needed. Additionally, subjects' insights concerning their experience corroborate the inferences. More measures are required to objectify such evidence e.g. structured probe tests.

5.6. Future Directions Concerning Assessment

In this section, some suggestions for future work concentrating on the assessment of object-oriented interfaces are given. The suggestions are derived from a consideration of the limitations outlined in the previous section and the findings of the studies.

The limitations imply that though a start has been made in the assessment of object-oriented interfaces and the findings have been interesting, there is a need for a greater range of studies to provide a broader set of results and to enable greater generalisation.

Studies which might be undertaken include:

i) A series of experiments on single features of object-oriented interfaces to encompass all the proposed features.
ii) Replication studies concerning both syntax and other features. The range of task, commands etc. should be increased.

iii) Studies exploring possible interaction effects between combinations of features.

iv) Studies based in domains other than that of common office tasks.

v) General assessment studies, similar to the observational study, but using a wider range of interfaces.

Other directions which might serve to complete the picture of ease-of-use include:

i) A replication of the first and second experimental studies under more realistic conditions (as in the third experiment) to find out whether the results were a product of the artificial setting.

ii) An investigation focussing on explaining the underlying reasons for the object advantage as evidenced in the third study.

6. Conclusion

The investigation presented in the thesis, comprising of both analytic and empirical activities, has been successful in augmenting the knowledge base concerning object-oriented interface. The investigation has culminated in two outputs:

i) a characterization of object-oriented interfaces;

ii) empirical findings concerning the assessment of the ease-of-use of object-oriented interfaces.

While limitations have been identified with both the outputs, they provide useful groundwork in a little researched area.

The characterization stream has identified a range of features which indicate object-orientation. Classifying interfaces in terms of these features enables object-oriented interfaces to be distinguished from other classes of interface (WIMP's; metaphorical interfaces; direct manipulation interfaces). The majority of these features have been shown to have consequences for the performance of the user. Additionally, it has been determined that object-orientation at the interface is independent of the programming language in which the interface is coded.
The assessment stream has lead to the following propositions:

- in terms of general measures, object-oriented interfaces are easier to use than non-object-oriented interfaces, as defined by the characterization scheme.
- with respect to syntax of the command sequence, orientation towards objects, i.e. object => action syntax, facilitates performance under particular combinations of conditions.

- factors which influence the usability of particular syntactic orderings include: input style; task structure and topic assignment (dependent on the nature of the tasks and the domain).

Clearly, there is scope to build on the work of both streams by further verifying and validating the features for characterization and extending the focus of the assessment to encompass further object-oriented features and combinations of features over a range of domains, tasks and device simulations.

The thesis hopes to demonstrate that the object-oriented interface is a robust concept which would be an interesting and useful topic for further human factors study.
APPLE FAMILY OF BUSINESS COMPUTERS, Apple advertising material.


BYTE, August 1981, SmallTalk Issue.


296


PC USER, December 1986 issue.


List of Appendices

<table>
<thead>
<tr>
<th>Appendix</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.</td>
<td>Questionnaire for Observational Study</td>
<td>304</td>
</tr>
<tr>
<td>B.</td>
<td>Instructions for Observational Study</td>
<td>307</td>
</tr>
<tr>
<td>C.</td>
<td>List of Interventions (Observational Study)</td>
<td>309</td>
</tr>
<tr>
<td>D.</td>
<td>Templates for Tasks (Observational Study)</td>
<td>311</td>
</tr>
<tr>
<td>E.</td>
<td>Lists of Errors Scored (Observational Study)</td>
<td>321</td>
</tr>
<tr>
<td>F.</td>
<td>List of Tasks for Experimental Study 1</td>
<td>331</td>
</tr>
<tr>
<td>G.</td>
<td>Instructions for Experimental Study 1</td>
<td>333</td>
</tr>
<tr>
<td>H.</td>
<td>List of Tasks for Experimental Study 2</td>
<td>334</td>
</tr>
<tr>
<td>I.</td>
<td>Instructions and Questionnaire for Experimental Study 2</td>
<td>338</td>
</tr>
<tr>
<td>J.</td>
<td>Matrix of Results for Exploratory Study</td>
<td>345</td>
</tr>
<tr>
<td>K.</td>
<td>List of Tasks for Experimental Study 3</td>
<td>346</td>
</tr>
<tr>
<td>L.</td>
<td>Instructions for Experimental Study 3</td>
<td>347</td>
</tr>
</tbody>
</table>
Appendix A

Questionnaire for Observational Study

QUESTIONNAIRE

This questionnaire asks for some personal details about yourself and for information about your previous experience with computers. All answers will be treated confidentially.

PERSONNAL DETAILS

Sex .................................................................
Date of Birth ...................................................
Nationality .....................................................
First Language ..............................................
Other languages spoken ...............................  

What kind of qualifications do you hold? (e.g. O Levels; Diploma; Degree)
.................................................................................................................................
.................................................................................................................................

ABOUT YOUR JOB

Present job title .............................................
Length of time spent in present job .................
Please give a brief description of the nature of your work:

........................................................................................................................................
........................................................................................................................................
........................................................................................................................................
........................................................................................................................................
........................................................................................................................................

Previous jobs held:

........................................................................................................................................
........................................................................................................................................
........................................................................................................................................

YOUR EXPERIENCE WITH COMPUTERS

Do you use computers at work? .................................................................

If yes,
How long have you been using them? .................................................................
How often do you use computers? .................................................................

Please give the names of the machines and software packages you use.

........................................................................................................................................
........................................................................................................................................
........................................................................................................................................

What sort of tasks do you use them for?

........................................................................................................................................
Do you use computer at home?

If yes,

How long have you been using them?

How often do you use computers?

Please give the names of the machines and software packages you use.

What sort of tasks do you use them for?
Appendix B

Instructions for the Observational Study

1. General Instructions.

I am conducting this study as part of my research for a Ph.D at University College London. I am interested in finding out what makes some computers easier to use than others. Therefore I am interested in people's performance generally rather than in your competence as an individual.

Today's session will be organised in three parts:

In the first part you will be given some training to familiarize you with the computer you are going to use. The remaining two parts form the test session when you will be asked to perform some general office tasks using the computer. During the test session a video recording will be made of your performance. Please do not worry about this, the film will be used for research purposes only. Your performance will be kept completely confidential and the film will be identifiable only by number.

Thank you for your co-operation.

Please ask if you have any questions or if you are unsure about anything.

2. Test Session Instructions: Part I

In the first part of the test session you will be asked to do a set of tasks using the computer. Please try to attempt these in as normal a manner as possible - as if you were doing an ordinary day's work. The tasks should take less than 45 minutes, try to do your best on these tasks but don't worry if you have difficulty completing them, you will be stopped after 45 minutes, if necessary.

If you wish to think aloud at any time, please feel free to do so.

If you have any serious difficulties please ask out loud, I shall be able to hear from the next room and I can answer over the speaker system.

Below there is a description of the task scenario.
Scenario

Imagine:

You work as an executive for a management services company. Your work involves liaising with colleagues as well as interacting with customers of the company.

The company has installed computers and all executives have one in their office. You use the computer for all your written work - writing and editing reports, letters, memos, filing information etc. You've just come back from your holidays and you have to catch up on all your correspondence.

Please work through the set of tasks in order.

If you have any questions, please ask.

3. Test Session Instructions: Part II

You will now be shown a video recording of your performance on the tasks.

I would like you to talk me through the tape, saying how you were attempting to do the task at each point. If you made any mistakes or had any difficulties I'd like you to try to explain what you thought about them and how you got around these difficulties. Please talk about your experience as fully as possible.
## Appendix C

### List of Interventions (Observational Study)

<table>
<thead>
<tr>
<th>Subject</th>
<th>Number of interventions</th>
<th>Task &amp; nature of advice required</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>Task 2</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- advice about opening the 'Disk' icon to recover from a 'Set Aside Everything command'</td>
</tr>
<tr>
<td>S1</td>
<td>1</td>
<td><strong>Task 2</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- advice about what to do about the information read.</td>
</tr>
<tr>
<td>S2</td>
<td>1</td>
<td><strong>Task 2</strong></td>
</tr>
<tr>
<td>S7</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>S8</td>
<td>4</td>
<td><strong>Task 2</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- advice about how to find the letter, i.e. access the letters in order and read them.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Task 4</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- reminder about the Current folder and its location.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- advice about how to open the Disk icon (after a 'Set Aside Everything' command)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Task 5</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- reminder about filing and moving icons by Direct Manipulation.</td>
</tr>
<tr>
<td><strong>TOTAL:</strong></td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>
Task 2
- advice about returning to the TopView directory.
- advice about how to access a file (using 'Type' command)

Task 4
- advice about pressing Mouse Button 1 to set cursor position.
- advice about the task: what to do next.
- advice on where to look for the Letter.
- advice on how to use the Personal Editor; where to type the file address etc.

Task 5
- advice about where to look for the memo.
- advice about selecting the cursor position again.
- advice about the exact filename to enter.

Task 4
- advice concerning the unavailability of the Personal Editor; too many DOS Services windows had been opened.
- reminder about the correct position for typing in the file address.
- reminder about accurately typing in the file address, i.e. not omitting the '/'.

Task 5
- reminder about the file name for copying.

Task 4
- question about doing the tasks in order and a reminder about the location of the letter.

TOTAL: 15
Appendix D

Templates for tasks (Observational Study)

This appendix contains templates showing the steps necessary to perform the tasks of the Observational Study using the most efficient method. Templates are given for both Lisa and TopView.

TASK 1

**Steps for Lisa**

1. Open Current
2. Drag Memo-87 (icon) to Wastebin (icon)

**Steps for TopView**

1. Scroll
2. Select Current
3. Erase Memo-87
TASK 2

**Steps for Lisa**

1. (Close Current)
2. Open Letters
3. Open letters (n= 1 to 6)
4. Note information
5. Close letter

**Steps for TopView**

1. Return to TV Dir by altering command line to C:\TopView\*. *
2. Select Letters
3. Type Letters (n=1 to 6)
4. Note information
5. Return DOS Services
TASK 3

Steps for Lisa

Open letters (n= 1 to 6) locate letter MS-1

Rename letter MS-1

Repeat 1 & 2 for letter MS-2

OR

Open letters (n= 1 to 6) locate letters MS-1 and MS-2

Rename letters MS-1 and MS-2
Steps for TopView

(already in Letters Directory)

Type Letters
(n= 1 to 6)
Locate MS-1 and MS-2

Rename MS-1
MS-2

Type Letters
(n= 1 to 6)
Locate MS-1

Rename MS-1

Repeat above steps for MS-2
Appendix D

TASK 4

Steps for Lisa

PART 1

Open Current folder

Open Ref-GH
Scroll - ascertain gaps

Open Info folder

Open documents (n= 1 to 3)
locate figures

Size windows of Ref-GH and Figures.
Decide on information

Select Ref-GH window
Type in Figures

Close Ref-GH

PART 2

Move Ref-GH icon to Letters folder
Steps for TopView

PART 1

Return to TV Dir
by altering command line to
C:\TopView\*.*

Select Current directory

Type Ref-GH

Return to TV Dir
by altering command line to
C:\TopView\*.*

Select Info directory

Type documents
(n = 1 to 3)
locate figures

TopView menu
Switch to Personal Editor

PART 2

Enter:
/TopView/Current

Edit Ref-GH

Make changes,
file and quit.

Return to TV Dir
by altering command line to
C:\TopView\*.*

Select Current

Copy Ref-GH to:
/TopView/Letters
/Ref-GH

Erase Ref-GH
(from Current)
TASK 5

Steps for Lisa

1. Open Current
2. Select Memo-JS
3. Pick Duplicate from File/Print menu
4. Move new copy to Letters folder
5. Rename copy of Memo-JS
6. Open Memo-PJ (copy of Memo-JS)
7. Edit, Save and Put Away
Steps for TopView

(already in Current directory)

- Copy Memo-JS: \TopView\Letters\ Memo-PJ
- TopView menu
  - Switch to Personal Editor
  - Specify \TopView\ Letters
  - Edit Memo-PJ
  - Make changes,
    File & Quit
    Personal Editor

OR

- Copy Memo-JS: \TopView\Letters\ Memo-JS
  - Rename Memo-JS

318
TASK 6

Steps for Lisa

1. Open Letters folder
2. Open Ref-Comm: size window to half a screen
3. Open Ref-WJ: size window to other half of screen
4. Compare lengths
5. Click on Ref-Comm
6. Alter date
7. Close Ref-Comm
8. Move Ref-Comm icon to Current folder
Steps for TopView

Return to TV Dir by altering command line to C:\TopView\*. *

Select Letters directory.

Type Ref-Comm
Return DOS Services

Type Ref-WJ
Return DOS Services

Decide Ref-Comm is longer

TopView menu - Switch to Personal Editor

Specify \TopView\ Letters

Edit Ref-Comm

Make changes, file & quit Personal Editor

Copy Ref-Comm: to \TopView\ Current\Ref-Comm

Erase Ref-Comm (from Letters)
Appendix E

Lists of Errors (Observational Study)

The profile will consist of a list of the errors made (where errors are defined as deviations from the shortest path through the task) as well as short explanations as to the possible cause of the error.

**TASK 1**

**LISA**

<table>
<thead>
<tr>
<th>Subject Numbers &amp; List of Errors</th>
<th>Explanation/Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S7;S8) Opening the memo</td>
<td>This is an unnecessary task step: it could be seen as checking behaviour. Not an error of interaction.</td>
</tr>
<tr>
<td>(S1;S7;S8) Opening the File/Print menu</td>
<td>The subjects were looking for a selecting the Memo-87 icon. Discard option in the menu. They all initially forgot about putting things in the wastebin.</td>
</tr>
<tr>
<td>(S8) Selected the Save &amp; Continue option from the File/Print menu.</td>
<td>An unnecessary step.</td>
</tr>
<tr>
<td>(S1) Opening the wastebin in order to put Memo-87 in.</td>
<td>This demonstrates a reluctance to move an icon to an icon in order to put something inside something else.</td>
</tr>
<tr>
<td>(S7;S8) Set Aside Current</td>
<td>----</td>
</tr>
<tr>
<td>(S1;S8) Trying to select shadowed icons</td>
<td>This could indicate a mis-understanding about the function of shadowed icons.</td>
</tr>
<tr>
<td>(S7) Having the wrong item selected: thus unexpected options appeared.</td>
<td>The bin was highlighted when the subject intended to use the Current window.</td>
</tr>
<tr>
<td>(S8 x2) Incorrect movements used for dragging, resulting in items being put in the wrong place.</td>
<td>This seemed to be a mouse control problem - possible nervousness over Direct Manipulation.</td>
</tr>
<tr>
<td>(S8) Trying to put the Current folder in the Wastebin.</td>
<td>Problems with distinguishing between object types - here a folder and a document.</td>
</tr>
<tr>
<td>(S8) Attempt at putting the Wastebin into the Document.</td>
<td>Perhaps a misunderstanding of the metaphor: or failure to infer the correct semantic properties from the icon.</td>
</tr>
<tr>
<td>Subject Numbers &amp; List of Errors</td>
<td>Explanation/Comment</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>(S4) Selection of incorrect action (Re-name).</td>
<td>This can be attributed to a mouse slip.</td>
</tr>
<tr>
<td>(S4) A number of attempts at selecting 'Erase'.</td>
<td>These were problems of mouse control.</td>
</tr>
<tr>
<td>(S5) Displaying the memo using 'Type'.</td>
<td>This was an unnecessary task step: possibly checking behaviour.</td>
</tr>
<tr>
<td>(S6) Failure to select Erase on first attempt.</td>
<td>Incorrect use of mouse.</td>
</tr>
<tr>
<td>(S5) Selecting Memo-JS instead of Memo-87.</td>
<td>Due to a mouse slip.</td>
</tr>
</tbody>
</table>

**TASK 2**

LISA

<table>
<thead>
<tr>
<th>Subject Numbers &amp; List of Errors</th>
<th>Explanation/Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S1) Mistakenly chose &quot;Set Aside Everything&quot;.</td>
<td>This error occurred twice, the second time it was a mouse error. The first time it could have been due to a confusion between Set Aside and Save.</td>
</tr>
<tr>
<td>(S1) Attempted to recover from the above by trying: - the menu - Current icon - Clipboard icon</td>
<td>These were attempts to either find the Letter or to recover back to the original position. &quot;Disk&quot; did not seem to be an icon that was easily recognizable. The second occurrence of this error caused no problems.</td>
</tr>
<tr>
<td>(S1) Attempt to select a shadowed icon.</td>
<td>A minor confusion over shadowed icons but the remedy was quickly remembered.</td>
</tr>
<tr>
<td>(S8) Opened and closed Letters.</td>
<td>This was an unnecessary step and may have been due to S8 not recognising the letters or to a more general problem about the nature of folders.</td>
</tr>
</tbody>
</table>
Appendix E

TOPVIEW

Subject Number & List of Errors | Explanation/Comment
--- | ---
(53) Failed to press button 1 to signal cursor position, when changing directory. | A case of forgetting minor part of procedure for changing directory.
(54) Failed to edit command line to change directory, tried: - Switch (x2) - Quit - Help | Method for returning to the TopView directory had been forgotten. Subject tried various strategies and then asked for help.
(S5 x2) Failed to edit command line to change directory tried: - Switch and selected a new DOS Services screen (reverts to C directory listing). | Again method for changing directory had been forgotten. Chose a new DOS Services from switch panel.
(S5) Searches through C directory for the letters. A consequence of the above error. | Subject did not realise that the directory listing was for the C directory, not TopView.
(S3) Accessed the TopView menu in the wrong place. | Here the button 3 was pressed when the pointer was in the command panel so the options were not the required ones.
(S4) Selected Copy instead of Type. | This was a problem of finding a suitable command for displaying the contents of a file.

TASK 3

LISA

Subject Numbers & List of Errors | Explanation/Comment
--- | ---
(S1) Selected "Attributes" from File/Print menu. | This was not necessary; however it was in keeping with the subject's strategy of looking for some sort of log file.
(S1) Selection of Save & Put Away Letters: contingent on above error. | The subject had to remove the Attributes panel and in so doing chose to close the Letters file, however this error was recovered.
(S1 x2) Moving icons inadvertently | This was an error of controlling the mouse.
(S1) Mis-spelling during process of renaming. | Typing error.
(S1) Several attempts at deleting the error generated as above. This subject found the Delete function very awkward - probably due to negative transfer from her previous experience.

(S7) Mis-selection of menu item. This was due to a mouse error - letting go of the button when the mouse was in the wrong position.

**TOPVIEW**

<table>
<thead>
<tr>
<th>Subject Number &amp; List of Errors</th>
<th>Explanation/Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S4) Incorrect input leading to Error message in Rename panel.</td>
<td>Due to a typing error - pressing Enter at the wrong time.</td>
</tr>
<tr>
<td>(S3, S5 in part) Scanning down the Dates column to ascertain dates of letters without opening them to check.</td>
<td>Not really an error, more a case of a different task strategy.</td>
</tr>
</tbody>
</table>

**TASK 4**

**LISA**

<table>
<thead>
<tr>
<th>Subject Number &amp; List of Errors</th>
<th>Explanation/Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S1, S7, S8) Opened the Info folder first.</td>
<td>This is a task error.</td>
</tr>
<tr>
<td>(S1, S2, S7, S8) Looking for Ref-GH in Letters</td>
<td>Again a task error - just happens that this is the wrong location. Subjects did not have problems in terms of interaction.</td>
</tr>
<tr>
<td>(S1) Repeated error above.</td>
<td>Forgetting location again - easily recovered.</td>
</tr>
<tr>
<td>(S1 x2, S7, S8) Inadvertent moving of icons.</td>
<td>Mouse error - not letting go of the mouse button when selecting.</td>
</tr>
<tr>
<td>(S1 x2, S8) Attempt to select a shadowed icon.</td>
<td>Failure to remember implications of shadowed icons.</td>
</tr>
<tr>
<td>(S1) Failed to move the insertion point during editing.</td>
<td>Forgot to press mouse button to signal position for insertion point.</td>
</tr>
<tr>
<td>(S1) Problem with Delete function</td>
<td>Due to negative transfer from previous experience as before.</td>
</tr>
<tr>
<td>(S8) Missed out two task steps i.e. did not reopen the report and make a note of the figures - just added fictional figures.</td>
<td></td>
</tr>
</tbody>
</table>

324
### TOPVIEW

<table>
<thead>
<tr>
<th>Subject Number &amp; List of Errors</th>
<th>Explanation/Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S3, S4, Selecting the Info S5, S6) directory first.</td>
<td>Task error: selecting the Info directory first is an inappropriate step.</td>
</tr>
<tr>
<td>(S6) Selected Rename by mistake</td>
<td>Mouse error</td>
</tr>
<tr>
<td>(S3, S4, Selected Letters S5, S6 x2) directory - searched through.</td>
<td>Task error - happens to be the wrong location for the letter: this is ascertained by looking through some/all of the letters.</td>
</tr>
<tr>
<td>(S3, S4 x2, S5) Typing error in editing command line (to change directory).</td>
<td>Problems with changing the command line.</td>
</tr>
<tr>
<td>(S5) Failure to correct above error</td>
<td>This lead to a listing of all items containing the string, 'TopView'.</td>
</tr>
<tr>
<td>(S4) Tried to correct above error with TopView menu</td>
<td>An inappropriate attempt at correction as switching cannot be used to swap between directories.</td>
</tr>
<tr>
<td>(S3, S4, Selected Letters S5 x2) directory again [after already locating the letter in Current]</td>
<td>The subjects forgot the location of the letter.</td>
</tr>
<tr>
<td>(S5 x2, S6 x2) Selected Info again.</td>
<td>Here the subjects forgot the information to be copied and had to re-check it.</td>
</tr>
<tr>
<td>(S5 x2) Selected a new DOS services when trying to change directory. This meant that TopView had to be selected from the C: directory.</td>
<td>Subject adopted this strategy: change directory for changing directory instead of altering the command line.</td>
</tr>
<tr>
<td>(S3, S4) Attempted to edit the letter from the DOS Services environment.</td>
<td>Subjects had the letter displayed on the screen and wanted to edit it immediately.</td>
</tr>
<tr>
<td>(S6) Selected Current directory again.</td>
<td>Subject had forgotten the name of the letter and the name has to be entered into the Personal Editor environment.</td>
</tr>
<tr>
<td>(S4) Errors made in switching to Personal Editor. - selected DOS Services twice. - tried to put ref-gh on the command line in DOS Services.</td>
<td>This first attempt was abandoned. Subject had problems - choosing the editor from the switch list.</td>
</tr>
<tr>
<td>(S5) Large problems encountered when trying to change to the Personal Editor. Too many DOS Services panels had been opened and not quit - this caused the memory to be too low for loading the</td>
<td>When the error message appeared the experimenter intervened and explained to the subject that some of the DOS Services would have to be quit. The subject was guided through rectifying the situation.</td>
</tr>
</tbody>
</table>
Appendix E

Personal Editor and an error message appeared.

(S4) Made errors filling in the Personal Editor details panel, but these were corrected.

(S4) In the Personal Editor entered the wrong filename tried: "gh" and 'ref gh" these lead to the New File message. Then the Edit command was omitted.

(S6) Missed out '\' on first attempt at filling in Panel for Personal Editor

(Could not remember exact format.

(S5) Typed the directory details in the wrong place in the Personal Editor details panel. This resulted in new file messages. Second attempt - missed out the '\'. Results in an error message, but subject continues on into the editor. This results in New File messages Correct on third attempt.

(S5) Entered the wrong filename in the Personal Editor - tried "Current" and "gh".

This seemed to be a case of forgetting the exact filename and also not distinguishing the file from the directory.

(S5) Tried to use the TopView menu for scrolling inside the Personal Editor.

Inappropriate - the subjects have been shown to use the cursor keys.

(S5) Tried to use the TopView menu to quit the Personal Editor.

Inappropriate quitting must be achieved using the function keys in the Personal Editor.

TASK 4 (PART II)

LISA

<table>
<thead>
<tr>
<th>Subject Number &amp; List of Errors</th>
<th>Explanation/Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S8) Accessed File/Print menu for some filing option.</td>
<td>Seemed to have forgotten about filing by using direct manipulation - looking for an action.</td>
</tr>
<tr>
<td>(S8) Tried to re-file the letter by moving the whole Current folder.</td>
<td>Some misconception here about what an icon represents. Folders are treated as documents.</td>
</tr>
<tr>
<td>(S8) Looked in File/Print menu (with letters selected).</td>
<td>Again an attempt at finding an option for filing.</td>
</tr>
<tr>
<td>(S8) Selected LisaGraph Paper</td>
<td>This was selected inadvertently: mouse error.</td>
</tr>
</tbody>
</table>
(S8) Looked in File/Print menu (with LisaGraph Paper selected).

(S8) Selected Set Aside Everything

(S7) Opened Letters

A further attempt at filing.

There seemed to be some equation between setting aside and filing.

Unnecessary step - an example of checking behaviour.

**TOPVIEW**

**Subject Number & List of Errors**

**Explanation/Comment**

S3) Mistakes in filling in the Copy Panel for the letter, "Ref-GH":
- misspelt TopView
- missed out TopView (but realised)
- missed out '
- put \TopView\Letter'

This resulted in the letter being copied to the TopView directory under the name of "Letter".

**TASK 5**

**LISA**

**Subject Number & List of Errors**

**Explanation/Comment**

(S7) Incorrect selection - duplicate option unavailable.

Due to a mouse error - easily recovered.

(S8) Document open so duplicate option unavailable (only available for icons)

Subject recovered.

(S7) Moving an icon inadvertently

Mouse error.

(S7) Problems with text cursor during renaming

Minor problems - recovered.

(S8) Set Aside Everything by mistake: several attempts to recover - failed to re-open disk.

The setting aside caused major problems as subject did not know how to open disk - various icons and menus were tried. An intervention was necessary.

(S8) Unnecessary Setting Aside; followed by putting item away.

Subject seemed confused over the functions of setting aside and saving and putting away.

(S8) Copied memo to Current which involved an extra step of moving copy after editing.

Subject had problems with moving and gave up trying to move it and concentrated on copying and
Appendix E

(S8 x2) Attempting to re-file folder instead of document. A common confusion for this subject, between a folder and the documents it contains.

(S8 x2) Looking in File/Print menu for filing option. Subject had problems with filing via direct manipulation: here looking for a menu option.

(S8) Sizing window difficulties. Largely due to mouse difficulties.

TOPVIEW

<table>
<thead>
<tr>
<th>Subject Number &amp; List of Errors</th>
<th>Explanation/Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S6) Returned to TopView and reselected Current - although she had been in Current already.</td>
<td>This was unnecessary: indicates that subject did not have a good idea about where she was in the system.</td>
</tr>
<tr>
<td>(S3) Copied Memo to TopView directory, renaming it Letter.</td>
<td>Incorrect specification of new location results in the file being copied to the wrong place. Subject was unaware of the error until the beginning of the next task.</td>
</tr>
<tr>
<td>(S3) Renamed the original memo.</td>
<td>Perhaps a mis-interpretation of the task instructions or a more general confusion of which file was which.</td>
</tr>
<tr>
<td>(S3) Missed out ‘\’ when trying to access the Personal Editor.</td>
<td>Minor error when filling in details.</td>
</tr>
<tr>
<td>(S3) Edited the original memo.</td>
<td>Again a mis-conception or confusion.</td>
</tr>
<tr>
<td>(S4 x3; S5 x2) Selected Letters directory when looking for memo.</td>
<td>This was a repeated error of looking in the wrong location.</td>
</tr>
<tr>
<td>(S4) Selected Print by mistake.</td>
<td>This was a mouse error.</td>
</tr>
<tr>
<td>(S4) In returning to TopView erased too much of the command line.</td>
<td>Typing error - recovered.</td>
</tr>
<tr>
<td>(S4) Copied the memo to the Current directory, calling it Letters.</td>
<td>An incorrect specification was given for the new location. Subject seemed confused about the path name convention - did not realise where the copy had gone.</td>
</tr>
<tr>
<td>(S4) A number of difficulties in specifying the file address in Personal Editor.</td>
<td>Subject did not have a clear idea of the correct address or of the address he had used.</td>
</tr>
<tr>
<td>(S4) In Personal Editor, omitted the Edit command.</td>
<td>Due to the above error new file messages appeared and on this attempt to input the correct filename S4 omitted the command.</td>
</tr>
<tr>
<td>Task 6</td>
<td>Subject Number &amp; List of Errors</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>(S4)</td>
<td>Put incorrect filename with</td>
</tr>
<tr>
<td></td>
<td>Edit command</td>
</tr>
<tr>
<td>(S4)</td>
<td>Specified file address</td>
</tr>
<tr>
<td></td>
<td>incorrectly in Personal Editor.</td>
</tr>
<tr>
<td>(S5)</td>
<td>A mis-selection.</td>
</tr>
<tr>
<td>(S5)</td>
<td>Cancelled the operation</td>
</tr>
<tr>
<td></td>
<td>Copy Memo-J5.</td>
</tr>
<tr>
<td>(S6)</td>
<td>Problem over command line when</td>
</tr>
<tr>
<td></td>
<td>changing directory.</td>
</tr>
<tr>
<td>(S6)</td>
<td>Unnecessary Type Memo-PJ</td>
</tr>
<tr>
<td></td>
<td>command.</td>
</tr>
</tbody>
</table>

**LISA**

**TOPVIEW**

<table>
<thead>
<tr>
<th>Task 6</th>
<th>Subject Number &amp; List of Errors</th>
<th>Explanation/Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S8 x3)</td>
<td>Sequence of opening and closing</td>
<td>These were unnecessary steps and seemed to constitute an attempt at filing.</td>
</tr>
<tr>
<td></td>
<td>windows.</td>
<td></td>
</tr>
<tr>
<td>(S8 x3)</td>
<td>Attempts to select shadowed</td>
<td>Problem over the function of icons.</td>
</tr>
<tr>
<td></td>
<td>shadowed icons.</td>
<td></td>
</tr>
<tr>
<td>(S8 x2)</td>
<td>Attempts at moving whole folder</td>
<td>A confusion over the properties of these objects. Document and the folder which contains it are used in a synonymous manner.</td>
</tr>
<tr>
<td></td>
<td>instead of document.</td>
<td></td>
</tr>
<tr>
<td>(S8)</td>
<td>Confusion over the functions of</td>
<td>The operations of these were not properly understood.</td>
</tr>
<tr>
<td></td>
<td>Set Aside and Save &amp; Put Away.</td>
<td></td>
</tr>
<tr>
<td>(S8 x2)</td>
<td>Moved icon to desktop before</td>
<td>Unnecessary.</td>
</tr>
<tr>
<td></td>
<td>filing.</td>
<td></td>
</tr>
<tr>
<td>(S8)</td>
<td>Icon moved inadvertently.</td>
<td>A mouse error.</td>
</tr>
<tr>
<td>(S7)</td>
<td>Deselected icon prior to opening</td>
<td>Deselection due to a mouse error - easily so options became inapplicable.</td>
</tr>
<tr>
<td></td>
<td>menu recovered.</td>
<td></td>
</tr>
</tbody>
</table>

**TOPVIEW**

**Subject Number & List of Errors**

<table>
<thead>
<tr>
<th>Task 6</th>
<th>Subject Number &amp; List of Errors</th>
<th>Explanation/Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(S3, S6) Did not type out the</td>
<td>Different task strategy employed by both subjects - S6 did type out one of the letters just to check the strategy.</td>
</tr>
<tr>
<td></td>
<td>two letters for a comparison of</td>
<td></td>
</tr>
<tr>
<td></td>
<td>length.</td>
<td></td>
</tr>
<tr>
<td>(S3)</td>
<td>Sort by size.</td>
<td>Extra innovative step to subserve task strategy.</td>
</tr>
<tr>
<td>-------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>(S3 x2)</td>
<td>Selected Print.</td>
<td>Mouse error</td>
</tr>
<tr>
<td>(S3)</td>
<td>Omitted '/' in Personal Editor panel.</td>
<td>Minor failure to use correct format.</td>
</tr>
<tr>
<td>(S3)</td>
<td>Omitted '/' in the Copy panel.</td>
<td>Minor failure to use correct format.</td>
</tr>
<tr>
<td>(S3)</td>
<td>Sort by name.</td>
<td>Extra innovative step.</td>
</tr>
</tbody>
</table>
Appendix F

Instructions for Experimental Study 1

1. General Instructions

I am conducting this study as part of my research for a Ph.D at University College London. I am interested in finding out what makes some computers easier to use than others. Therefore I am interested in people's performance generally rather than in your competence as an individual.

Today's session will be organised in two parts:

In the first part you will be given some training to familiarize you with the computer you are going to use. The second part is the test session in which you will be asked to perform a set of tasks using the computer. During the test session a video recording will be made of your performance. Please do not worry about this, the film will be used for research purposes only and performance will be kept completely confidential.

Thank you for your co-operation.

Please ask if you have any questions or if you are unsure about anything.

2. Test Session Instructions

Introduction

As you are probably already aware the use of computers in the work place is becoming increasingly common. Often this means that people whose work is not directly related to computers; use computers to assist them in managing information. Computers are often used to support common office tasks such as writing and editing reports; sending messages; accessing information which is centrally stored in a data-base etc. Tasks such as these are performed by a wide range of professional workers irrespective of the area of their expertise (e.g. lawyers, accountants, business managers, advertising executives, lecturers etc.) I am particularly interested in looking at the use of the computer for such tasks.
Test Session

In the test session you will be presented with a series of short tasks. These are the sorts of tasks you might expect any office worker to perform as part of their daily work. Normally, of course the worker would know which tasks needed to be done and would not have to be instructed.

A description of each task will appear on the screen. Please read each description as quickly as necessary for you to gain an idea of what you have to do. As soon as you have finished reading, click on the OK button. Then you will be ready to start the task. Please do each task as quickly and as accurately as you can. You can only have one attempt at each task.

If you have any questions about the test session, please ask.

Some additional information was provided for the Command Line Groups. The information given to the object -> action group is reproduced below:

At the beginning of each task you will have to type in a command sequence and press "Enter". Each command sequence will have two words.

For example:
if you wanted to look at the calendar you would type:

    calendar display (Enter)

if you wanted to change something in a report you would type:

    report edit (Enter)

Please do each task as quickly and as accurately as you can. You can only have one attempt at each task.

If you have any questions about the test session, please ask.
Appendix G

Task Instructions for Experimental Study 1

The task instructions which appear below are those which were given to subjects in the Action/Pick & Point Group (AP); half the subjects received these instructions and half received them with the ordering of the objects and actions reversed.

Task 1  Copy the memo because you have decided to keep a record of all memos
Task 2  The directory should be sorted so that the entries are displayed by department
Task 3  Send the report to your colleague Ms S Jones
Task 4  The memo should be edited so that your new phone number reads 4321
Task 5  The mailbox should be displayed so that you can find out how many messages you have received
Task 6  Print out the calendar so that you have a record of what's happening in July
Task 7  Send the memo to Mr H Francis to inform him of your new phone number
Task 8  The report should be copied because you want to file all company information for future reference
Task 9  Sort the calendar to show the dates of forthcoming meetings
Task 10 Edit the graph so that it takes the form of a pie-chart
Task 11 The mailbox should be printed out so that you have a reminder of all your messages
Task 12 The calendar should be displayed so that you can find out whether there is a conference in June
Task 13 Copy the graph so that you can use it to produce future graphs
Task 14 The directory should be printed out so that you can use it at home
Task 15 The report should be edited so that the interest rate mentioned in Section C reads 5.75%
Task 16 Display the directory so that you can find out the number for D F Langham
Task 17 The graph should be sent to your boss, Mr Green, for his approval
Task 18 Sort the mailbox so that the messages are listed in order of their date
Appendix H

Tasks for the Second Experimental Study

Examples of tasks used in the first test session of the second study are reproduced below.

Letter
Display to find out the name of the colleague who is going to accompany you to the meeting. Then, send a copy to the colleague who will accompany you. Make a standard-size, best-quality print-out to put in the post to W J Hethering. Now that you've finished working on it file it away with your other letters.

Graph
You need to edit two of the figures to agree with the amended figures which you have just received in a message: A should be 46 and L should be 27 - then re-plot. The name is no longer appropriate, so rename to read, "Amended Sales Figures Oct. 88". You need a copy to file elsewhere, call it, "New Figs". Compact to take up as little space as possible.

Chart
As it consists of explanatory material for a project, this should be filed under, "Documentation". You need to check out whether you have included any input from the Market Research Agency in your scheme, so display to find out. The lab. team manager just called to say that they now intend to test out 8 recipes so you will have to edit to incorporate this information. All the people involved in the project want to have a look at your plan, so make 8 draft quality print-outs to hand round at the next meeting.

Memo
You want to re-use this memo, so make a copy of the original, called "Memo/Storage". Now you've had an enquiry about the market research material from D M Cherry, edit the name of the recipient and the date so it is ready to send to him - then send it to him. You like to group together all material of an organisational nature under the heading "Admin", put the memo under this heading.

Message
You've realised that you made a spelling error in the name - rename it to read "Message / Davison". Find out the date of this message. If it was sent after 5th September, send it on to your colleague, Mick Fleet because he says he hasn't received anything since early September. If the message currently takes up more than 8 units, compact it.

Report
You think the name is a bit too long, rename it with something more snappy like, "New Product Trials". You want to keep a copy of the report at this stage of its preparation, call the copy "Draft - 1.1".
You want to conserve disk space so reduce the space that the report takes up. Eventually the report will be published, so file it away under "Publications".

Record-card

You want to use this as an illustration for a presentation so you will need an overhead projector foil - make a best quality, enlarged print-out for this purpose.
Send to your manager, H. Brown, as he has asked to have a look at it.
You want to give it a new name, "De-luxe Range - Apricot Crumblies".
Later you want to produce cards like this for all the products in the range, so make a copy called "Prototype" - you can make new cards from it when you are less busy.

Send

You have been promising to send out some copies of things to various people:
Ms R Derwent in Publications needs to have a look at the table.
The memo should go to Mr J Allthorpe as he frequently uses Market Research Material.
K Jones, the department administrator has asked for the graph so that she can distribute it to all the people on the mailing list.
Your project plan needs approval from the project leader, Chris Carter before you can go ahead, he needs to see the chart.

Display

You need to find out a few bits of information from things that you have written recently.
Yesterday someone queried the serial number for Apricot Crumblies - look up the record-card to check it.
Check where the material from Jeeves research is kept - the memo has details about it.
You have a moment of doubt about the graph - did you include a figure for each of the 15 products?
Look in the report to see if there is any information missing you will have to amend it later.

Edit

You've found out the name of the Product that was missing from the report, fill in the gap with "Coffee Snap".
Deadlines have slipped a little and now it doesn't look like you'll have the figures until mid-December, alter the second paragraph of the letter accordingly.
Make a note to yourself about the figures in the message - put "check figures for other regions" to the right of the new figures.
The prices on the record-card need to be changed to: Retail = 85p and Wholesale = 49p.

Print

You need two draft quality print-outs of the memo to stick up in noticeboards around the department.
The report is going to be bound with the quarterly reports, so you should make a best quality print-out which can be dispatched to Publications.
The graph is also going to be included in a publication so make three best quality, enlarged print-outs.
Make a draft quality print of the message to show to the Production Team.

File

You have decided to put everything in sensible places, so that you can find them again!
Put the message under "Communications"; where all your other correspondence is kept.
All pictorial information is stored under the heading "Graphics", so include your graph under this
heading.
The record-card is going to be useful for meetings so put it under the heading, "Meetings".
The sales information is usually kept under "Products"; this seems a good place to keep the table.

Rename

You think that some of your names are too cumbersome, shorten that of the letter to read, "WJH/Letter".
The memo should really be called "Memo/Storage of Market Research Material".
Your manager left you a note earlier saying that the new product is going to have the code, A54321:
so you need to alter the name of the chart to read "Proposed Plan for Product A54321".
You'd prefer to call the table "Figures for the Savoury Range 1988".

Compact

The letter is rather long so you need to condense it to save disk space - see if you can compact it by at least 25%.
Try this on the chart if you find that it takes up more than 6 units.
See which takes up more space - the table or the memo; make them both smaller.

TEST SESSION 2

The Action-type tasks for test session 2 are reproduced below. The Object-type tasks used these sub-tasks, but grouped them according to their object.

Print

You need a best-quality print out of the letter to put in the post this afternoon.
You want to make an overhead projector foil from the table so produce a best-quality hard copy.
The Packaging team have asked for a couple of prints of the record-card; so make some best-quality print outs.
The memo will make a useful reminder of deadlines so make a draft quality, reduced size, print-out to pin on your noticeboard.

Send

Your colleague, P James, has asked to see the graph before you discuss figures with her this afternoon.
Sue Jones is going to give you her ideas for the next stage of the report so she needs to see your draft.
You'd like some feedback concerning your presentation so let your manager, H Brown, have a look at the record-card.
Show the memo to the department administrator K Jones, so that she can incorporate the information in the Dept. diary.

Compact

You are aware that your available disk space is getting low so you need to check the sizes of things and reduce them if necessary.
You've decided to reduce the letter if it takes up more than 6 units.
You want the table to be as economical as possible.
The report should not be bigger than 8 units.
Messages don't usually take up much space, check this one and then reduce it.

Rename
You've decided to use shorter names for memos so rename it, "Ads./Memo".
You've just received a memo saying that the product number for Sesame Crackers has now changed to A3378 - you'll have to alter the name of the record-card accordingly.
The report is not yet finished so you think a more appropriate name would be, "Draft Report -Advertising", so change it.
Before you show the graph to a colleague you have to rename it, "Sales Figures for All Regions (UK)".

Display

Look in the letter to find out the reference number of the report that you have requested: the library already has a report numbered JR6778-C so it might not be necessary to send this letter.
Have a look at the chart to see whether you have included the people from London and Milton Keynes in the Management team.
You've suggested possible dates for a meeting in the memo - there's another meeting proposed for 8th November - check whether there is a clash.
Check to see how recent the message is - you have an idea it was sent after the end of September.

Edit

Some new figures have just arrived, fortunately there's only one that is important for you; it means you'll have to alter point N of the graph to read 58.
The deadlines have now been altered - the advertising strategy must be completed by the end of Jan, so the bids must be in by the end of December, edit the chart to reflect this.
Also, you have just realised that the figures for the North shown in the table are, in fact, only for the North-East, so change "North" to "North-East".
The product number for Sesame Crackers has changed to A3387; change the Serial Number on the record-card to correspond with the new product number.

File

As the project described by the chart is in an early stage, there are numerous meetings about it, so put the chart with your other information concerning meetings.
You like to put all incoming information together in a file called "Communications" - include the message in this file.
As both the graph and the record-card contain specific product information you want to put them with other information of the type under the heading "Products".

Copy

You want a copy of the report called, "old version", because you will soon make a lot of amendments and you'd like to keep a record of all previous drafts.
You'd like to make a copy of the letter which should be called, "KFC/Letter".
Make a copy of the chart which you can edit if necessary to show an alternative plan, call the copy "Chart 2".
You have received the message fairly recently so make a copy called "Deadlines" - then you can incorporate it into your project plan sometime.
Appendix I

Instructions and questionnaire for second experimental study

Instructions

I am conducting this study as part of my research for a Ph.D at the Ergonomics Unit, University College London. I am interested in finding out what makes some computers easier to use than others. Therefore I am interested in people’s performance generally rather than in your competence as an individual.

Today’s session will be organised in three parts:

In the first part you will be given some training to familiarize you with the computer you are going to use. The second part is a test session in which you will be asked to perform a set of tasks using the computer. The third part is also a test session in which you will be asked to do another set of tasks. During both test sessions a video recording will be made of your performance. Please do not worry about this, the film will be used for research purposes only and performance will be kept completely confidential.

Afterwards you will be asked to complete a questionnaire concerning your thoughts about the computer you used in the test sessions.

Thank you for your co-operation.

Please ask if you have any questions or if you are unsure about anything.

Test Session

Introduction

As you will no doubt be aware, computers are becoming increasingly common in the work place. These days many people whose work is not directly related to computers use computers to assist them in managing information. For example, many retail managers use them to keep track of sales and stock; booking agency clerks and travel agents use them to manage information about bookings; many designers use them to produce drafts of their designs etc.

Often computers are used to support common office tasks such as writing and editing reports; sending messages; accessing information which is centrally stored in a data-base etc. Tasks such as these are performed by a wide range of professional workers irrespective of the area of their expertise (e.g. lawyers, accountants, business managers, advertising executives, lecturers etc.). I am particularly interested in looking at the use of the computer for such tasks.
In the test sessions you will be asked to do a number of office tasks using the computer.

Imagine you are an executive working in the Sales and Marketing Department if a large biscuit manufacturing firm. Fairly recently the company has installed computers and each executive has his or her own machine and is expected to use it to support his or her daily work.

As an executive you have many on-going tasks such as writing up reports; setting up meetings; making presentations etc. and you organise your own time around the various deadlines for these tasks.

Of course, each of the tasks incorporates a number of sub-tasks, for example setting up a meeting might involve:-

- sending memos to prospective attendees
- preparing an agenda
- booking a room
- contacting people from outside the company by 'phone or letter
- preparing material to be presented at the meeting etc.

Many of these sub-tasks could be supported by a computer e.g. memos could be sent electronically to people within the company; an agenda could be typed in and printed out; telephone numbers could be found from a directory stored on computer etc. These are the sort of tasks which you will be asked to do in the test session. (Of course normally, the worker would know which tasks needed to be done and would not have to be instructed).

Computer System Description

This is a description of the computer you are going to use.

The computer contains the following items:

- **Chart**: A diagrammatic/tabular description of some information
- **Diary**: A paged listing of day-to-day plans; appointments etc.
- **Directory**: A listing of all the internal 'phone numbers used in the company
- **Graph**: A graphical description of some data (usually numeric)
- **Letter**: A text document, specifically designed for communicating
- **Memo**: A short text document, specifically designed for communicating with other members of the company
Appendix I

Message
A text document which has been sent from someone from another company site

Record-card
A short card which gives brief information (textual and/or pictorial) about a particular subject

Report
A longer length textual document

Table
A tabular depiction of information, using numerical and textual data

and the following things can be done to them:

Archive
To put something that you won't be needing to work on for some time into long-term storage

Compact
To condense something so that it take up less storage space on the disk

Copy
To make an electronic duplicate of something

Display
To show something on the screen

Edit
To make modifications (including substituting; adding and deleting) to the contents of something

File
To store something under a particular named classification

Print
To make a hard copy of something

Rename
To change the existing name of something

Retrieve
To relocate a previously archived item to the current workspace

Send
To transmit something to someone (within the company) in an electronic form
Specifying Commands

To begin a task you have to specify a command sequence by clicking on the items you want. Each command sequence consists of two items and they have to be selected in a particular order e.g.

If you wanted to look in the diary you would select:

    Diary       Display

If you wanted to change something in the graph you would select:

    Graph       Edit

If you select items in the wrong order then a message appears in the message area (like you saw in the training).

Test Session II

Now for the second test session.
This session is almost the same as the last one, but you will be using a slightly different computer.

This new computer looks the same as the one that you have been using. The only difference is that the command sequence can be entered in any order:

e.g. if you want to look at the diary you can enter:

    Diary       Display

or

    Display      Diary

The computer accepts both versions.

Now attempt the tasks in the same manner as before, choosing the command sequence that suits you best.
Questionnaire

Section 1

Please could you fill in the following personal details and information about your previous computer experience.

Sex ____________ Nationality ____________

Date of Birth ____________ Right-handed/ Left-handed ____________

First Language ____________ Other Language ____________

What kind of qualifications do you hold (e.g. A Levels; Diploma; Degree etc.) ?

__________________________________________________________________________

What is your occupation (job title / course name, if student / unemployed) ?

__________________________________________________________________________

Have you used a computer before ? ______________

If yes, was it at work or at home ? ______________

How long have you been using a computer ?

__________________________________________________________________________

How often do you use one (daily; 2-3 times a week; once a month etc.) ?

__________________________________________________________________________

What types of computers and packages have you used (e.g. IBM PC; BBC; WordStar; Microsoft Word etc.) ?

__________________________________________________________________________

What sort of tasks have you used a computer for (e.g. wordprocessing documents; statistics;
programming; games; data-base enquiry etc.)?

Have you ever used a mouse before?

Section 2

This section asks some questions about your reactions to the test sessions which you have just done. Please circle the answer which is most appropriate for you.

1. Did you find the computers easy to use?
   YES
   NO

2. Which computer did you find easiest?
   TEST 1
   TEST 2

3. Did you have any difficulty in using the mouse?
   YES
   NO

4. Did you find the command words easy to understand?
   YES
   NO

5. Do you think a computer like this would be useful?
   VERY USEFUL
   QUITE USEFUL
   A BIT USEFUL
   NOT USEFUL

Some questions about the first test session:

6. Did you ever choose the command words in the wrong order?
   YES
   NO

7. Did you find it difficult to use the order that was required?
   YES
   NO
8. Did you like the order that you had to use?

YES

NO

9. Try to write down the command words for the computer you have just used:

_________________________________________________________________
_________________________________________________________________
_________________________________________________________________
_________________________________________________________________
_________________________________________________________________

Some questions about the second test session:

10. Did changing to a different computer make a difference to you?

VERY MUCH

A BIT

HARDLY AT ALL

11. Which order did you prefer to use?

OBJECT -> ACTION

ACTION -> OBJECT

MIXED

Please make any additional comments below:

_________________________________________________________________
_________________________________________________________________
_________________________________________________________________
_________________________________________________________________

Thank you very much for your help
Matrix of Results (Pilot Study)

<table>
<thead>
<tr>
<th></th>
<th>card-index</th>
<th>calculator</th>
<th>calendar</th>
<th>diagram</th>
<th>folder</th>
<th>in-tray</th>
<th>diary</th>
<th>telephone</th>
<th>letter</th>
<th>notepad</th>
<th>memo</th>
<th>filing-cabinet</th>
<th>clock</th>
<th>graph</th>
<th>wastebin</th>
<th>directory</th>
<th>address</th>
<th>book</th>
</tr>
</thead>
<tbody>
<tr>
<td>not</td>
<td>3</td>
<td>10</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>9</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>7</td>
<td>9</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>send</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>1</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>type</td>
<td>9</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>look in</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>pick up</td>
<td>6</td>
<td>9</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>9</td>
<td>9</td>
<td>6</td>
<td>0</td>
<td>3</td>
<td>8</td>
<td>0</td>
<td>9</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>copy</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>8</td>
<td>5</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>pet</td>
<td>10</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>8</td>
<td>5</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>10</td>
<td>0</td>
<td>7</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>consult</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>switch on</td>
<td>9</td>
<td>8</td>
<td>4</td>
<td>6</td>
<td>9</td>
<td>7</td>
<td>5</td>
<td>9</td>
<td>1</td>
<td>4</td>
<td>10</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>3</td>
<td>10</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>throw away</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>7</td>
<td>9</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>9</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>enlarge</td>
<td>6</td>
<td>7</td>
<td>2</td>
<td>7</td>
<td>5</td>
<td>3</td>
<td>8</td>
<td>1</td>
<td>8</td>
<td>9</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>dial</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>re-label</td>
<td>7</td>
<td>7</td>
<td>1</td>
<td>2</td>
<td>10</td>
<td>9</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>0</td>
<td>8</td>
<td>7</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>empty</td>
<td>2</td>
<td>7</td>
<td>3</td>
<td>4</td>
<td>9</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>reply</td>
<td>3</td>
<td>8</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>7</td>
<td>10</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>3</td>
<td>10</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>draw</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>9</td>
<td>9</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>file</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>edit</td>
<td>10</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Appendix K

Task Instructions for third experimental study.

i) Test Instructions (computer tasks)
(as communicated over the telephone)

1. Can you find out William Evans' phone number for me? - I'll hang on.

2. Your information concerning the names, addresses and services of various suppliers needs to be arranged alphabetically.

3. You've got a lunch appointment with Ted Carpenter at 12.30 on Wednesday 26th July - mark it down so that you don't forget it.

4. You need to let Ted know that you can make the meeting at lunchtime on the 26th.

5. I need a set of your figures to take to a meeting this afternoon. Can you get them ready for me and I'll pick them up on my way.

6. We need to write to two more of the candidates for the job - to invite them to a second interview. We'll send the same thing that was sent to the first person, so if you could just produce another two - we'll finish them off later.

7. The Sales Manager has plotted the sales figures for all the restaurant - you should put this with the rest of your Sales Information.

8. All the material that has been amassed concerning New Projects is now out of date, so discard it.

ii) Non-computer tasks
(as on the TO DO list)

1. Proof-read the report on the findings of the Market Research Questionnaire - mark all errors in RED (it will have to be re-typed).

2. Fill out expenses claim forms for the recent batch of interviewees - one has been done already.

3. Sort your personnel index cards into groups, one for each restaurant. Make sure that within the groups the cards are put alphabetically.

4. Check the Sample menu for mistakes and add any comments or ideas for alternative dishes.
Appendix L

Instructions & Scenario Description for E3

1. General Instructions

I am conducting this study as part of my research for a Ph.D at University College London. I am interested in finding out what makes some computers easier to use than others. Therefore I am interested in people's performance generally rather than in your competence as an individual.

Today's session will be organised in two parts:

In the first part you will be given some training to familiarize you with the computer you are going to use. The second part is the test session in which you will be asked to perform a set of tasks using the computer. During the test session a video recording will be made of your performance. Please do not worry about this, the film will be used for research purposes only and performance will be kept completely confidential.

Thank you for your co-operation.

Please ask if you have any questions or if you are unsure about anything.

2. Details about the Test Session

During the test session please imagine that you are the executive described in the scenario and that you are working through some of your everyday tasks . . .

In the test session you will be given a folder and a list of things to do - all the relevant pieces of information for these tasks will be in the folder. Just work through these tasks in your own way - trying to be as efficient and accurate as possible.

While you are working your 'phone will ring a number of times. When you answer I will give you some instructions. Please attempt to carry out these instructions straight away - leaving the task that you are doing aside. In order to carry out the instructions you will have to use your computer. The instructions which I give you may seem a bit vague - please interpret them as best you can. I won't be able to answer questions or give you fuller details on the 'phone. However, don't worry, all the computer tasks are fairly simple.

After you have finished carrying out the instructions just return to the task you were doing before the 'phone rang.

Please ask if you are unsure about anything.

Good luck!
3. Scenario:

You work as an executive for a Management Company which specialises in managing restaurant chains. Your work concerns a chain of Italian Pizza/Pasta Restaurants in London. At present there are eight restaurants in the chain; the company would like to expand in the near future and is hoping to set up new restaurants in other parts of London and the South-East.

Your department is called "Group T". The department has five executive workers, four juniors and one administrator. The department is often involved in the setting up of new restaurants. Recently a new restaurant was successfully set up in the Docklands area. The new restaurant is called "Alfredos" and it has been running for just over a year.

Currently you are working on two main tasks:

- recruiting new staff for Alfredos: you have just conducted interviews for the positions of chef and waiter/ress.

- advising and co-ordinating a new project to set up a restaurant in Kent (after the initial stages this project will be undertaken by the Group P department). This aspect of your work involves you in many meetings with the new project group.

Your company has recently installed computers into all its offices. You are in the process of transferring your work to the computer. You expect that the computer will be very useful in supporting your everyday tasks - things like diary keeping; writing letters; storing and organising information; producing graphs etc.

At the moment some of your tasks can be done using the computer while others are still done in the usual way.