Applicability of HCI Techniques to Systems Interface Design

Victoria Mary Elizabeth Bellotti

1990

Thesis Submitted for Examination for the Degree of Doctor of Philosophy

Registered at Queen Mary and Westfield College, London
Applicability of HCI Techniques to Systems Interface Design

Abstract

This thesis seeks to identify reasons why HCI techniques are unsuitable for application in real world design projects. User-oriented systems design and evaluation require that many considerations such as the psychology of users, the applications and target tasks be born in mind simultaneously. A selection of influential HCI design and evaluative techniques from HCI research literature are reviewed and characterised in terms of their analytic scope.

Two studies of systems designers' approaches to user-oriented design and evaluation were carried out in order to gain a clearer picture of the design process as it occurs in applied and commercial projects. It was found that designers frequently lack adequate information about users, carrying out, at best, informal user-evaluations of prototypes. Most notably HCI design and evaluative techniques, of the type common in the literature, are not being used in applied and commercial design practice. They seem to be complex, often limited in scope, and possessed of inadequate or unrepresentative views of the design process within which they might be applied. It was noted that design practice is highly varied with only a small number of common goal directed classes of activity being identified. These together with observed user-oriented information sources and design constraints provide a useful schema for viewing applied and commercial design practice.

A further study of HCI specialists' practice in commercial environments was undertaken, in order to identify particular user-oriented design approaches and HCI techniques suitable for application in practice. The specialists were able to describe desirable, and undesirable properties of the techniques they used which made it possible to identify a list of specific desirable features for HCI techniques. A framework for assessing applicability of HCI techniques was developed from the findings of the thesis. This is demonstrated using an example project from the design studies and may prove valuable in supporting design, evaluation, critiquing and selection of HCI techniques.
Applicability of HCI Techniques to Systems Interface Design

Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page No.s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgements</td>
<td>17</td>
</tr>
<tr>
<td>Executive Summary</td>
<td>18</td>
</tr>
<tr>
<td>Chapter 1</td>
<td>22</td>
</tr>
<tr>
<td>Evaluating Usability With Respect to HCI Principles</td>
<td>22</td>
</tr>
<tr>
<td>1.1 System Usability and Evaluation Within Design and Not in a Vacuum</td>
<td>22</td>
</tr>
<tr>
<td>1.2 Evaluation Factors and Principles of Usability</td>
<td>26</td>
</tr>
<tr>
<td>1.2.1 Evaluation Factors</td>
<td>26</td>
</tr>
<tr>
<td>1.2.2 Incorporating Principles of Usability in HCI Approaches</td>
<td>30</td>
</tr>
<tr>
<td>1.2.3 Compromising Principles</td>
<td>36</td>
</tr>
<tr>
<td>1.3 HCI Models</td>
<td>38</td>
</tr>
<tr>
<td>1.3.1 HCI Design and Evaluative Techniques</td>
<td>39</td>
</tr>
<tr>
<td>1.3.2 Psychological and Interaction models</td>
<td>40</td>
</tr>
<tr>
<td>1.3.3 Competence and Performance Models</td>
<td>41</td>
</tr>
<tr>
<td>1.4 Questions of Appropriateness and Applicability for HCI Approaches</td>
<td>43</td>
</tr>
<tr>
<td>1.4.1 The system user(s); An Individual or a Population</td>
<td>45</td>
</tr>
<tr>
<td>1.4.2 The Nature of the System Application(s)</td>
<td>46</td>
</tr>
<tr>
<td>1.4.3 The System Interface</td>
<td>47</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>1.4.4 Target Task(s)</td>
<td>48</td>
</tr>
<tr>
<td>1.4.5 Acceptable Interactive Performance</td>
<td>49</td>
</tr>
<tr>
<td>1.5 Summary</td>
<td>50</td>
</tr>
<tr>
<td>Chapter 2</td>
<td>52</td>
</tr>
<tr>
<td>A Review of HCI Design and Evaluative Techniques</td>
<td></td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>52</td>
</tr>
<tr>
<td>2.1.1 Overview</td>
<td>52</td>
</tr>
<tr>
<td>2.1.2 Classifying and Characterising the Scope of HCI Techniques</td>
<td>53</td>
</tr>
<tr>
<td>2.1.3 Usability Principles With Respect to Evaluation Factors</td>
<td>55</td>
</tr>
<tr>
<td>2.2 HCI Models</td>
<td>67</td>
</tr>
<tr>
<td>2.2.1 The Block Interaction Model (BIM); A Model of Models</td>
<td>68</td>
</tr>
<tr>
<td>2.2.2 Reisner's Formal Interactive Grammar</td>
<td>73</td>
</tr>
<tr>
<td>2.2.3 Task Action Grammar</td>
<td>79</td>
</tr>
<tr>
<td>2.2.4 ACT*</td>
<td>87</td>
</tr>
<tr>
<td>2.2.5 Goals Operators Methods and Selection-Rules (GOMS)</td>
<td>93</td>
</tr>
<tr>
<td>2.2.6 Cognitive Complexity Theory</td>
<td>104</td>
</tr>
<tr>
<td>2.2.7 Interacting Cognitive Subsystems</td>
<td>119</td>
</tr>
<tr>
<td>2.2.8 Command Language Grammar</td>
<td>129</td>
</tr>
<tr>
<td>2.3 Summary</td>
<td>144</td>
</tr>
<tr>
<td>Chapter 3</td>
<td>147</td>
</tr>
<tr>
<td>HCI Techniques in UI Design and Evaluation: The Theorist's View and The Practitioner's View</td>
<td>147</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>147</td>
</tr>
</tbody>
</table>
Chapter 4
A Features Analysis of Design Projects:
Designing Systems for Users

4.1 Introduction and Rationale 201

4.2 Questionnaire Structure and Methodology 202
4.2.1 Structure of the Questionnaire 203
4.2.2 Distribution of the Questionnaire 204
4.2.3 Analysis 206

4.3 Basic Findings from Descriptive Responses to Part I 207
4.3.1 Grouping Projects According to Host Organisation 207
4.3.2 Project Size 208
4.3.3 Familiarity with HCI 211
4.3.4 Product Functions and the Type of UI Involved 213
4.3.5 The Prospective Users 214
4.3.6 Organisations involved 216
4.3.7 Information from Requirements, and Design Specifications, Task Descriptions and Users 217
4.3.8 Structured Specification Techniques and Descriptive Methodologies Used for Requirements, Design and Evaluation 222
4.3.9 Generation and Evaluation 223
4.3.10 Problems and Modifications 225
4.3.11 Finalisation of the Project 230
4.3.12 Lessons Learned 232
4.3.13 Constraints on Design Activity 233
4.3.14 Information Sources 240
4.3.15 Subjective Evaluations of Good and Bad Design Features 250

4.4 Discussion of Results 251
4.4.1 Grouping Projects According to Host Organisation 251
4.4.2 Project Size 253
4.4.3 Familiarity with HCI 253
4.4.4 Product Functions and the Type of UI Involved 254
4.4.5 The Prospective Users 254
4.4.6 Organisations Involved 255
4.4.7 Information from Specifications, Task Descriptions and Users 256
4.4.8 Specifications and Methodologies Used 257
4.4.9 Generation and Evaluation 258
4.4.10 Problems and Modifications 258
4.4.11 Finalisation of the Project 259
4.4.12 Lessons Learned 260
4.4.13 Constraints on Design Activity 260
4.4.14 Information Sources 261
4.4.15 Subjective Evaluations of Good and Bad Design Features 263

4.5 General Discussion 264

4.6 Conclusions 269

Chapter 5 270
An Interview-Based Investigation of Applied and Commercial Design Practice Activities and Problems

5.1 Introduction and Rationale 270

5.2 Interview Structure and Methodology 275

5.3 Findings 276

5.3.1 Eight Design Scenarios 277
5.3.2 Projects Summary 286
5.3.3 Categories of Design and Development Activity 289
5.3.4 Commercial Design Problems 295
Prototypes and or Products

6.4.2 Technology Transfer 362
6.4.3 Reactive System State and Dynamic Behaviour Modelling 366
6.4.4 Task Analysis for Knowledge Based Descriptions 374
6.4.5 Command Language Grammar 382
6.4.6 Summary 388

6.5 General Discussion 389

6.5.1 Working Regimes, Activities and Roles 389
6.5.2 Important Features of Applied HCI and User-Oriented Design and Evaluation Techniques 392

6.6 Conclusions 404

Chapter 7
Systems UI Design and HCI Techniques: Presenting a View of Current Design Practice as it Relates to HCI

7.1 Overview and Introduction 409

7.2 Resume and Application Requirements of HCI DETs from Chapters 2 and 3 410

7.2.1 Block Interaction Models 410
7.2.2 Reisner's Formal Interactive Grammar (FG) 411
7.2.3 Task Action Grammar (TAG) 413
7.2.4 ACT* 414
7.2.5 Goals Operators Methods and Selection-Rules (GOMS) 415
7.2.6 Cognitive Complexity Theory 417
7.2.7 Interacting Cognitive Subsystems 418
7.2.8 Command Language Grammar 420
7.2.9 Common Attributes of HCI DETs 421
7.2.10 General Application Requirements of HCI DETs 424
7.3 Review of Main Findings from the Design Studies in Chapters 4 and 5 and their Implications for HCI DETs

7.3.1 The Features Analysis of Design Practice
7.3.2 The Supplementary Qualitative Study of Design Practice
7.3.3 Contrasting Design Evidence with Application Requirements
7.3.4 Summary of Implications of Design Practice for HCI Techniques

7.4 Conclusions

7.4.1 Problems of Inapplicability
7.4.2 Problems with Confidence
7.4.3 Assessing Applicability

Chapter 8 Discussion of the Use of HCI Approaches in Design Practice and a Framework for Assessment of HCI DETs for Application to Design Practice

8.1 Overview and Introduction

8.1.1 Design Practice With or Without HCI
8.1.2 Acting upon the Failure of HCI DETs to Penetrate Commercial Design

8.2 Implications of the Study of Commercial HCI Practice for Current HCI DETs

8.2.1 Overview of Findings: General Observations of HCI Practice
8.2.2 Overview of Findings: Reasons for Use and Non-Use of HCI DETs
8.3 Ideal Properties Required of Applicable HCI Techniques.

8.3.1 Limited Investment or Major Pay-Offs
8.3.2 Accuracy
8.3.3 Expressiveness, Clarity and Communicability
8.3.4 High Generality and Broad Scope
8.3.5 Successful Exploitation of Information
8.3.6 Integratability
8.3.7 Support for Analyst
8.3.8 Some General Points

8.4 A Classification of Specific Desirable Features of HCI DETs

8.5 AF: An "Application Framework" for Assessment of the Applicability of HCI DETs

8.5.1 The AF Empirically Based Commercial Design Schema
8.5.2 The AF Usability Scoping Matrix
8.5.3 The AF HCI Roles and Activities Matrix
8.5.4 The AF Desirable Features List
8.5.5 Possible Roles of the Application Framework

8.6 Illustration of Application Framework’s Descriptive Power

8.6.1 An Example of Application Framework: CCT and the Workstation Window Manager
8.6.2 Summary of the Application Framework for Assessing HCI DETs

8.7 Summary

Chapter 9
General Summary and Conclusions

9.1 Overview 526

9.1.1 Summary of Chapters 1 to 8 526
9.1.2 Summary of Contributions of the Research to HCI 529
9.1.3 Shortcomings of the Research 531

9.1.4 Related Work on Scoping of HCI DETs 531a

9.2 Summary of Implications for HCI DETs 532

9.3 Recommended Further Research 537

References 539

Appendix 1 554
Appendix 2 568
Appendix 3 578
Appendix 4 585
Appendix 5 589
Applicability of HCI Techniques to Systems Interface Design

List of Tables and Figures

<table>
<thead>
<tr>
<th>Table or Figure and Heading</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 2.1: Scoping Matrix of Classes of Usability Principle Over Evaluation factors</td>
<td>56</td>
</tr>
<tr>
<td>Figure 2.1: A Generic Block Interaction Model</td>
<td>70</td>
</tr>
<tr>
<td>Table 2.2: Usability Scoping Matrix for BIM</td>
<td>71</td>
</tr>
<tr>
<td>Table 2.3: Usability Scoping Matrix for Reisner’s Action Language</td>
<td>77</td>
</tr>
<tr>
<td>Table 2.4: Usability Scoping Matrix for Task Action Grammar</td>
<td>85</td>
</tr>
<tr>
<td>Figure 2.2: A General Framework for the ACT Production System, Identifying the Major Structural Components and their Interlinking Processes</td>
<td>91</td>
</tr>
<tr>
<td>Table 2.5: Usability Scoping Matrix for Anderson’s ACT*</td>
<td>94</td>
</tr>
<tr>
<td>Figure 2.3: The Basic Architecture for the MHP from Card, Moran and Newell (1983)</td>
<td>97</td>
</tr>
<tr>
<td>Table 2.6: Usability Scoping Matrix for GOMS</td>
<td>103</td>
</tr>
<tr>
<td>Figure 2.4: Example of Selection Rules in a Job Representation for Editing a Manuscript</td>
<td>107</td>
</tr>
<tr>
<td>Figure 2.5: Goal Structure for a Secretary Selecting Device for Editing a Manuscript</td>
<td>108</td>
</tr>
<tr>
<td>Figure 2.6: Goal Structure for Deleting a String of Text</td>
<td>109</td>
</tr>
<tr>
<td>Figure 2.7: State Transition Network for a Delete String Function</td>
<td>113</td>
</tr>
<tr>
<td>Figure 2.8: Example of a Task-to-Device Mapping between the Device Hierarchy and the User’s Goal Hierarchy for Delete String</td>
<td>114</td>
</tr>
<tr>
<td>Table 2.7: Usability Scoping Matrix for CCT</td>
<td>117</td>
</tr>
<tr>
<td>Figure 2.9: An Architecture for Perception, Cognition and Action (from Barnard 1985)</td>
<td>122</td>
</tr>
</tbody>
</table>
Table 4.13: Incidence of Constraints Experienced by Respondents 237
Table 4.14: Ranked Importance of Constraints Experienced by Respondents 238
Table 4.15: Incidence of Information Sources Exploited by Respondents 242
Table 4.16: Ranked Importance of Information Sources Exploited by Respondents 244
Table 4.17: Good and Bad Features of UI Designs Described by Respondents 252

Table 5.1: Problems Experienced in Interface Design and Projects Where They Were More Apparent 298
Figure 5.1: Framework for Common User-Oriented Design-Cycle Activities Based Upon Interviews with Designers 303
Table 5.2: The Ten Most Important Design Constraints Identified in the Features Analysis of Design Practice 306
Table 5.3: Design Constraints and Problems Combined from the Features Analysis and the Designer Interviews 310
Table 5.4: The Ten Most Important Information Sources Identified in the Features Analysis of Design Practice 313
Figure 5.2: Schema for User-Oriented Design Practice: Organisation, Constraints and Information Sources Based Upon Findings of Two Empirical Studies 319

Table 6.1 HCI Relevant Higher Educational Qualifications of HCI Specialists 328
Figure 6.1: HCI Skills Within a Group Structure 330
Table 6.2: Characterisation of Variation in Working Regimes of Organisations in the Study 339
Table 6.3: Summary of Activities and Roles

Table 6.4: Qualities of Activities Involved in the Various Roles of HCI Specialists in the Study

Table 6.5: Roles of HCI Specialists in the Four Organisations in the Study

Table 6.6: Usability Scoping Matrix for Experimental User Evaluation

Table 6.7: Usability Scoping Matrix for the Technology Transfer Technique

Figure 6.2: A Very Simple Statechart

Table 6.8: Usability Scoping Matrix for Statecharts

Table 6.9: Usability Scoping Matrix for TAKD

Table 6.10: Usability Scoping Matrix for CLG

Table 6.11: Support of Scientific Techniques Identified for Various Activities of HCI Specialists in the Study

Table 7.1: The Eight Projects Included in the Qualitative Design Practice Study

Table 7.2: Scoping Matrix of CCT Contrasted with Analytical requirements of Project 5 from the Designer Interview Study

Table 8.1: Overview of the Application Framework for Assessment of Practical HCI DETs (AF)

Figure 8.1: Schema for User-Oriented Design Practice Organisation, Constraints and Information Sources Based Upon the Findings of Two Empirical Studies

Table 8.2: Expanded Scoping Matrix of Classes of Usability Principle Over Evaluation

Table 8.3: Matrix of Support Requirements for Activities Involved in the Various Roles of HCI Specialists in the Study
Table 8.4: Desirable Features of Applicable HCI DETs 503

Figure 8.2: EX Schema for Design Practice Organisation, Constraints and Information Sources Based Upon Project 5 from the Designer Interview Study 507

Table 8.5: Expanded Scoping Matrix of Classes of Usability Principle Over Evaluation Factors 512

Table 8.6: List of HCI-Oriented Applied and Commercial Design Practice Activities from Chapter 6 514

Table 8.7: Matrix of Support Requirements for Activities Involved in the Various Roles of the Designers in EX Including Assessment of CCT Support 515
Acknowledgements

The author wishes to thank the following people whose contributions have been welcomed:

Dr Peter Johnson, her supervisor for his time and for the guidance he has given in the conduction of the research reported in this thesis, and for his further efforts during its preparation.

Dr Allan Maclean for his generosity during the period of writing-up the thesis, and for his useful advice in its final stages.

Dr Steve Sommerville for his assistance and support throughout and also for clarifying a number of problematic issues.

Dr Paul Walsh for his inspirational encouragement.

Eliot Miranda for support, advice and tolerance.
Executive Summary

This thesis seeks to address the utility of some influential HCI design and evaluative techniques for design practice in applied and commercial environments. It outlines the nature and scope of eight techniques and their relationship with other views of UI or user-oriented design practice. It should be noted that Chapter 2 which reviews these techniques need not be read in detail by those familiar with them.

Three studies, one based on questionnaires, and two based on interviews with designers and HCI practitioners, are reported which provide evidence which suggests that there are many problems with existing HCI techniques which make them difficult or impossible to apply in practice. The findings from the studies suggest a number of features which are required of ideal techniques. These features are developed within a framework for assessing the applicability of HCI techniques which may be used to support improvement in future techniques and to suggest where such techniques might be applied by systems designers or HCI practitioners.

Chapter 1

This chapter introduces the basic themes of the thesis which are to consider what the market for HCI is, and what perspectives (termed evaluation factors) need to be born in mind when designing and evaluating for usability of computer systems. It introduces a number of usability principles and describes types of HCI approaches which attempt to ensure that they are present in UI designs. Finally the implications of the various evaluation factors for systems designers are illustrated since they are likely to dictate what considerations are important for the UI, and hence will tend to suggest the types of issues which an HCI design or evaluation technique with appropriate scope would have to address.

Chapter 2

This chapter reviews eight influential HCI design and evaluative techniques from a largely theoretical point of view in order to give a clear general picture of their type, the methods they employ and the scope of the issues they address. Their scope is characterised using a "scoping matrix" which is based upon the discussions in the preceding chapter, and which gives a useful summary of the principles of usability
which they address and the evaluation factors which they account for.

Chapter 3

This chapter considers and criticises design views presented by HCI design and evaluative and how these compare with other user-oriented approaches to design. Empirical evidence relating to the application of HCI in systems design practice is also reviewed. The discrepancies between the various design views raise a number of serious questions relating to the applicability of HCI techniques of the type reviewed in Chapter 2. These questions are the main basis for the three studies reported in the following chapters.

Chapter 4

A questionnaire based study of applied and commercial design practice is reported in this chapter. Respondents were asked to describe design activities they carried out and various impressions they had of the design process for a single project which they had participated in, concentrating on the user-interface. They were also asked to describe problems they had experienced. Lists of possible design constraints and design activities were presented, and respondents were asked to rank those which they had experienced in approximate order of importance.

The study yielded a great deal of quantitative information which tended to suggest that the design process is varied, typically problematic and subject to a number of constraints. It also showed that informal approaches to UI design and evaluation are preferred, and that HCI DETs of the type described in Chapter 2 do not seem to be used in applied and commercial projects.

Chapter 5

This chapter describes a supplementary interview-based design study which was carried out in the light of the questionnaire-based study in order to provide a broader more contextualised view of user-oriented design practice in applied and commercial projects. The interviews, each focusing upon a single design project revealed some commonalities between design projects in the goals toward which design activities appeared to be directed, and a number of problems experienced by
designers which obstructed attempts to ensure usability of interfaces.

The findings elaborate on and tend to agree with those of the features analysis of design practice. The two studies together combine to provide a schematic view of typical applied and commercial design practice which involves certain types of activities, user-oriented information sources and design constraints.

Chapter 6

A third study is reported in this chapter which contrasts with the previous two studies in that it focuses upon the practices of, and techniques applied, by six HCI specialists from four commercial organisations. The data collection method was again interviews, but this time each one focused upon techniques applied across any number of projects rather than what happened during a single project.

The HCI practitioners provided a great deal of detailed information about the conditions under which they worked, and about the techniques they used. They were also able to describe the good and bad features of these techniques which suggest why they were used by the specialists and what their problems and limitations are. Notable amongst the findings was the fact that two HCI techniques of the type reviewed in Chapter 2 were being applied by the HCI specialists in one organisation.

Chapter 7

This chapter provides an overview of the HCI DETs reviewed in Chapter 2 in terms of the application requirements which they have. These requirements have to be met by design projects in order that the HCI DETs be applicable. The findings from the questionnaire- and interview-based design studies reported in chapters 4 and 5 are discussed in terms of the implications they have for the applicability of HCI DETs. The questions raised in Chapter 3 regarding the applicability of HCI DETs are answered on the basis of the empirical findings of these two studies.

Chapter 8

The findings of the HCI practitioners' interviews reported in Chapter 6 are overviewed here and, in the light of the findings from the two user-oriented systems
design practice studies, used as the basis for a set of features which seem to be required of applicable HCI DETs. These features are included within an empirically based "Application Framework" for assessing the applicability of HCI DETs. The framework contains three other components; an empirically based applied and commercial design schema (based upon the studies reported in chapters 4 and 5); a usability scoping matrix which is an extension of the matrix introduced in Chapter 2; an HCI roles and activities matrix which is drawn from the HCI practitioners study in Chapter 6.

The Application Framework is demonstrated on an example project from the designers' interview study in Chapter 5 and a number of possible roles for the framework are outlined.

Chapter 9

This chapter summarises the contents of the preceding chapters and their contribution to HCI. It also summarises some shortcomings of the research. Some general implications of the research for HCI DETs are discussed and, finally, further research in the area of applicability of HCI DETs is recommended.
Applicability of HCI Techniques to Systems Interface Design

Chapter 1

Evaluating Usability With Respect to HCI Principles:

In this chapter the problem of application of HCI to design practice is introduced. Design practice represents a context within which HCI techniques must work, which means that these techniques must be adapted to the demands of that context. Furthermore the techniques themselves must also consider a number of different factors which are introduced in this chapter, and which have important bearing on the usability of a system. These factors, which are derived from considerations which impact usability, define the context of interactions between users and systems and any evaluation of usability which does not consider them risks being meaningless.

A number of usability principles are also introduced which can be viewed as ideal properties of UIs which are desirable regardless of the context of interaction or the nature of the particular system in question. A variety of types of HCI technique have been developed to address these properties in design and evaluation. However, for the system designer the evaluation factors are likely to suggest that analysis supported by some types of technique might be more crucial than that supported by others since issues which are related to these factors (such as training, and the nature of tasks supported) vary widely between systems.

1.1 System Usability and Evaluation
Within Design and Not in a Vacuum.

Human-Computer Interaction (HCI) is a relatively applied science, devoted to studying the communication between people and computers. It tackles both physical and behavioural properties of humans and computer systems, and is multi-disciplinary, incorporating, amongst other things, ergonomics, psychology, and psycholinguistics. It is assumed that study of the nature of interaction between users and computers will lead to a better understanding of how to make computers easy to use.
Human interaction with computers is mediated by the user interface (UI) which can be thought of as the software and hardware aspects of the system that govern its observable behaviour and interpret inputs to the system. When we refer to system usability, we are generally talking about the physical and behavioural properties of the UI, such as the screen lay-out, the names of commands and the legal command sequences. Usability is also strongly related to the functionality of the system; that is the user's tasks which the application takes over (the term application is used to refer to the underlying software of the system, as opposed to the UI software; for example the application software would deal with creation, manipulation and deletion of data whereas the UI software would deal with how to accept input or display data). It is extremely difficult to provide a definition of the UI which specifies where the UI ends and where the rest of the system begins. In some sense, all aspects of the functionality of the system affect the UI. For the purpose of our discussion, it is assumed here that the necessary functionality of the system may be predetermined and largely outside the influence of those wishing to improve its usability. The UI is assumed to be something which mediated between that functionality and the user which can be improved with respect to usability, without affecting the overall system functionality. However this is not to say that altering functionality does not affect usability.

Maher & Bell (1977) speak of the UI as representing the virtual machine for the system user. Pylyshyn, (1984) states that the description of the virtual behaviour of a system can be thought of as distinct from descriptions of its underlying properties, whether the machine is a brain or a computer. This implies that we need not concern ourselves too heavily with the structure of the software when studying the UI, just as a biologist need not understand much chemistry to describe the physiology of animals. A further distinction can be made between necessary and interactive UI-behaviour. Necessary UI behaviour may be strongly dictated by its system application and hardware. It might manifest itself as delays in response times; the number (dictated by the user) and the resolution (dictated by the pixels of the screen) of characters used in text being edited in a word processing package. These aspects of the UI are not essentially designed as part of the UI itself. Interactive UI-behaviour is, hopefully, explicitly designed to mediate communication to and from the user, given the constraints imposed by the necessary behaviour, for example; a graph showing the levels of fluid in a tank over a period of time; highlighting of current selections; icons representing files or functionality. In the following, the term
usability is assumed to relate, on the whole, to the efficiency of interactive UI-behaviour, as opposed to business effectiveness or enjoyability which are perhaps outside the main focus of HCI at present.

Given, that much of HCI concentrates upon the UI and a single user's relationship with a system, and that it is reasonable to do so, then work in the field can be expected to yield focused and useful information about the properties of system UIs which make them easier to use... or can it? There have been a number of recent publications suggesting that the philosophy and techniques of HCI do not seem to be having the impact which HCI's claimed benefits would suggest it should (e.g. Hammond et al 1983, Gould & Lewis 1985, Grimes et al 1986, Rosson et al 1987). Although many techniques seem to have been developed, according to the claims of their authors, explicitly for the purpose of evaluating UIs and even designing them (e.g. Moran 1981, Card et al 1983, Kieras & Polson 1985, & Payne & Green 1986), there are sparse, if any reports of their actual use in design practice (Bellotti 1988).

It is interesting to note that this problem of the penetration by theoretically based approaches into actual practice, is not peculiar to HCI. For example, a very similar attack has been made on Information Systems (IS) design approaches by Lyytinen (1987):

"One reason for the abundance of IS design approaches is that it is quite easy to develop a method, but difficult to get it accepted.

"We argue, however, that the claim of a fundamental deficiency of systems development methodologies has not been taken sufficiently seriously to lead to a thorough and critical investigation of the basis on which current development methodologies build. There are few, if any convincing studies that verify the efficacy of proposed approaches. In fact we shall argue that proposed development approaches may even add to the crisis of IS failure. The reason for this is that they focus on a limited spectrum of development issues. Further their assumptions about the nature of systems development conflict with several empirical findings of its true nature."

Lyytinen goes on to argue that there are a number of specific weaknesses in IS development approaches which make them unacceptable or inapplicable. This
thesis represents an attempt to do a similar job for HCI approaches, but given the specialist nature of HCI as opposed to overall systems development, the demands upon such approaches are rather different to the demands for IS design approaches. For the latter, according to Lyytinen, the pressures are towards synergy with other approaches, greater scope, a wider conceptual base, broader theoretical foundations, and greater awareness of the philosophical underpinnings of systems development. Perhaps such pressures have led to the development of IS techniques with a broader set of perspectives, such as Multiview (Wood-Harper et al 1985) and SSADM (e.g. Downes et al 1988). However the generality of these approaches means that it is difficult to cover each aspect of the design in as much detail as specialist approaches recommend.

On the other hand, the scope of many HCI approaches is unlikely to extend beyond the design of the virtual machine and how that is mediated to the user, so the pressures on these approaches are different. Therefore such concerns as data abstraction, data encapsulation, core process design, parameterisation, and many other software issues are not considered here. However there are certainly some aspects of the software which do impact upon users (i.e. necessary interactive behaviour) even if they are not specifically designed to do so. It would seem unrealistic to suggest that specialist HCI approaches to design do not have to address the type of problems encountered by other systems design approaches, both specialist and general. Even if their scope is limited, they must address the fact that they will have to work in practice with approaches which deal with other areas of design.

A central theme in this thesis is therefore the need for HCI researchers to consider the market for HCI and its potential users, as systems designers must for their system. At present an inadequate view seems to be presented by HCI techniques of the scope of important system properties and factors which temper their assessment, or evaluation (e.g. Green et al 1987, Grudin 1989). An even less adequate view of the design process within which these techniques might be applied is provided with many techniques suggesting unusual and complicated activities be carried out (e.g. Kieras and Polson 1985) or providing no view of an appropriate design approach at all (Payne and Green 1986).

In this chapter a set of perspectives on Uls which may be termed evaluation factors and usability principles are presented. These are desirable properties which a system
design may possess, and factors which temper evaluation (and therefore must be
design considerations themselves). The description of these perspectives is neces-
sary since, in the following chapter, an attempt is made to outline the scope of a
number of influential HCI techniques which are reviewed.

1.2 Evaluation Factors and Principles of Usability

1.2.1 Evaluation Factors

The concept of usability cannot be captured by appealing to features of a (UI)
irrespective of the environment within which it exists (Cockton 1987a, Goransson et
al 1987). It is necessary to appeal to "features of the device, tasks, interaction
medium, and user knowledge that are important in determining the ease of use of a
support environment" (Green et al 1987). An interface to a system has to be sensi-
tive to the needs of its users who may be a special group with particular require-
ments. It must have appropriate behaviour in order to represent the information the
user requires about the behaviour of the application and must accept appropriate
input (such as mouse movements or menu selections) for the task in question. It
must also be within the price range of the purchaser and the limits of hardware and
software sophistication possible in the circumstances; in other words it is unlikely to
be the best UI available and compromises and optimisation will be necessary.

An ever widening variety of users (O.A.Ps to school children) interact with a
diverse range of computers (in the video recorder, and in the CAD workstation) via
an enormous range of Uls (from the automatic bank teller machine push-button UI,
via more stereotypical keypad-and-mouse-and screen through to the simulation of a
Boeing 757 cockpit).

The definition of system application(s) is made to provide a clear distinction
between the nature of the computer and the uses to which it is put which may or
may not be catered for by the applications. The system application or applications
may not be fully exploited by the users, and they may be abused by being used to
support tasks that they were not intended to originally. It is therefore important to
have a clear picture of both the application(s) and the intended uses required of a
design.
In a meaningful evaluation of a UI design at least three initial factors (putting aside cost for the time being) must be crucial in determining whether any given feature of the interface is to be judged usable. Without considering each of these factors to some extent, design could well turn out to be misguided and an evaluation is likely to be uninformative. The term evaluation factor which will be used from here on to refer to these, is assumed to have equal import for perspectives on design and evaluation. The three factors so far mentioned are:

1. The system user(s) with their particular experience, or lack of it, and their variability; an individual or a population = U(s)
2. The nature and function of the system application(s); the virtual machine, or the structures and behaviour which the system manifests, depending upon its hardware and software. = SApp
3. The system interface itself; which comprises the interaction medium = UI

From the above, usability may be said to be a system feature when:

\[ U(s) \times SApp \times UI \rightarrow Acc(TTs) \]

Where two further evaluation factors have to be considered:

4. TTs = target task(s); the set of target tasks which the user has to carry out, and which the system must support (a subset of the real world of possible manipulations of objects).

5. Acc = acceptable interactive performance (metrics may be time, errors, learning, successfully accomplished tasks, etc); matching or exceeding some performance criterion or criteria.

The above factors (1 to 5) may be considered as UI evaluation factors (EFs). Any attempt at evaluating an interface without considering these will be at best inadequate and at worst meaningless. They are introduced here to highlight the context within which interactions between users and systems take place. Even if only one of these is ignored, such as the application, the results of design or evaluation may be useless. For example the complaint that users would find a push-button UI to a large computer system very constraining, ignores the fact that for bank service tills such a UI has proved very successful. It must therefore be assumed that any realistic evaluation technique which does not, at least implicitly, deal with these issues,
must rely on the application of another technique which does, in order to yield valid results.

The natures of user, application and UI EFs can only be approximated to. Typically models of users and of the UI perhaps with some application behaviour also are built by HCI researchers, as these, for a number of reasons, are usually simpler and easier to study or evaluate. For example, Anderson (1983) has produced a model of the human information processing processes which has enabled him to simulate language learning, but which has also been used to support CCT (Kieras and Polson 1885) which is a technique for assessing the complexity of a UI for its human users. Similarly, RAPID-USE (Wasserman 1985) and Trillium (Henderson 1986) use State Transition Diagram based systems for simulating UI interactive behaviour in UI design approaches which facilitate early evaluations.

The target-tasks and acceptability of performance EFs are variables which can be specified, with respect to users, applications, and UIs, by the person(s) making the judgements as to whether the system is usable or not (for the sake of argument we will call the judge the analyst). However the threshold of acceptability and the set of target tasks may vary depending upon circumstances and the values of the analyst.

The conundrum of usability is then that it is often difficult to quantify. It is dependent on its context and the priorities of the analyst. In other words; whilst one analyst may judge an interface to be usable, another may find it less so, and if any factor(s) are manipulated, then both may change their mind. This makes the practice of improving this ill defined concept of usability extremely complicated in some circumstances.

We must always bear in mind that, with respect to usability, there is more to consider than simply the standard human cognitive architecture and the UI behaviour. Users' experience and background may be important, and the nature of the application and target tasks also have some bearing on evaluation, and therefore all of these should be considered in design also. The evaluation factors referred to above are intended to be used to preserve this breadth of scope of a viewpoint in any approach to design, be it oriented towards design or evaluation. By doing so it may be easier to avoid the type of narrow perspective which leads some researchers to focus upon
one aspect of the system to the extent that their methods fail to address obvious problems with that aspect, for example consistency; a popular HCI focus, has been shown by Grudin (1989) to interfere with usability when it is pursued whilst the users, their experience, their tasks, the nature of the UI and so on, have not been given proper consideration.

We need some consistent ideas about what the attainment of usability involves. It is attractive to go direct to representative potential users and carry out trials of the UI, but this approach entails several setbacks. Firstly the UI has to be in a state where users can actually get the look and feel of it, or at least see pictures of the proposed screen sequences and layouts. By this time many of the important decisions about the design probably will have been irreversibly finalised. Another problem is that user trials are often expensive, and the results they produce tend to mean more expense on making alterations. This is not to say that the expense is not worth it, in fact many studies (e.g., Gould and Lewis 1983, Mantei and Teory 1988, Rosson et al 1987) suggest that the cost of such trials is amply justified. However, most manufacturers and software developers may not be convinced by the evidence that the initial expense will be recouped later on.

Designers need to be guided in their designs by a reasonably precise set of ideas about what features of UIs are desirable within given contexts. Given that almost every design project involves new untested ideas and the problems arising are likely to be different for each project, what appears to be required are some consistent principles which can be successfully applied across user populations, applications and UI types to improve the UI design.

Given that usability is neither a certain, nor an easily defined property of a system, we shall assume that increased usability is a quantitative improvement in human performance with the UI, in terms of approximation to Acc(TTs), which can be approached through the application of certain principles. These principles are empirically supported, explicit hypotheses about what specific features characterise a more usable system. The disciplines of Human Computer Interaction (HCI) and Ergonomics both attempt to define such principles. These principles may be derived from psychological theories relating to possible system characteristics or properties, or descriptions of ideal design practice. They may be used as design goals or as standards for evaluation.
Landauer (1987) reviews some of the empirical, psychological research which has led to the generation of such principles. He discusses the difficulties of understanding "what matters in realistic contexts" which is where most UIs tend to have to prove themselves. Empirical psychological research yields only a fragmentary picture, in terms of the factors which underlie a user's ability to successfully interact with a system. Many different theoretical principles may have to be brought to bear upon any one aspect of performance in a given context. This means that the job of evaluating a UI (i.e. predicting the performance of a user) requires the analyst to successfully integrate a large number of principles. Such integration results in what is typically referred to as a model of some aspect of the interactive behaviour or the rules that govern it.

Thimbleby (1985) describes how principles should not be confused with slogans, guidelines and prejudices:

* Guidelines are applied principles, that is, principles worked out for a particular context.

* Principles are highly specific in the sense that, for slightly different circumstances, they may indicate slightly different approaches.

* Principles which are applied retrospectively (for kudos) are slogans.

* Prejudices (sometimes called "design features") are overgeneralised principles.

* Slogans are used for selling.

1.2.2 Incorporating Principles of Usability in HCI Approaches

Principles are constant, application and UI independent, and may be embodied in design guidelines. On the other hand, they may also be implicitly or explicitly embodied within models. A model may represent the system-user, the set of rules the user builds in order to interact with the system, the set of rules the system requires the user to learn for system interaction, and so on. HCI and psychological models are concerned, for the main part, with the behavioural properties of the system and its user. This thesis concentrates on those principles which are embodied in
such HCI models as have been developed to date.

There are a great many design and evaluation principles advocated by researchers and practitioners in HCI. These principles may be quite detailed but broadly speaking they can be grouped under the following more general categories of principles (The issue of modelling raised in the following section will be addressed further on).

Simplicity Principles
Compatibility Principles
User Centred Task Dynamics (UCTDs) Principles
Consistency Principles
Observability Principles
Retrievability Principles

The principles of simplicity, consistency, observability and retrievability are based upon extensive research by Dix and Harrison (1985), Monk and Dix (1987), Thimbleby (1984), and others which examined aspects of UI design which could be formally specified and which when embodied in a system yielded better usability. These principles when embodied in a UI have been shown to improve user performance by, amongst other things, reducing the amount of learning and remembering a user has to do, and improving feedback and error retrieval. Since these desirable properties could be formally defined they can, in theory, be embodied and evaluated in early system specifications without having to resort to expensive user testing and rewriting of software.

The principles of compatibility and user-centred task dynamics (UCTDs) are based upon research carried out by Barnard (1987), Carroll & Mack (1985), Moran (1981), Payne & Green (1986), Wilson et al (1988), Young (1983), and many others which has shown that ease of use of a UI depends to a great extent on the degree to which the concepts and rules required by users to operate it map onto their existing knowledge. In other words, at a number of levels of analysis, it has been shown that users' experience and expectations have an important role to play in determining their success or difficulty in using or learning to use a system. UIs which embody the principles of compatibility and UCTDs will be less likely to cause their users difficulty because the symbols, concepts, and rules they embody will permit people
to transfer existing knowledge to the new system.

Woods and Roth (1988) state; cognitive engineering must address the contents or semantics of a domain, since purely syntactic and exclusively tool-driven approaches to develop support systems are vulnerable to attempting to solve the wrong problem. Compatibility and UCTDs deal with semantics in that they address the relationship between the concepts and behaviour of the system and the representations held by the user. If these principles are embodied in the design of a UI, the user should find less difficulty with the terminology, concepts, and mental and physical operations required to complete tasks using the system.

In the following, a more detailed definition of each of these principles is given.

1. Simplicity

Simplicity is used here to refer to the number of operations, rules, (Anderson 1982b, 1983) and "families of rules" (Payne, 1984) that a user would have to represent in order to interact successfully with a system. Systems which have apparently complex behaviour and rules, but are easy to learn, may in fact be capitalising upon users' existing knowledge. Specific principles which might be included within the class of simplicity relate to such things such things as shortness of command-strings and using as small a number of operations as is possible to interact with and control the required functionality. UI simplicity might be improved, at the same time as ensuring a more flexible, modifiable design, by following suggestions made by di Sessa (1985). di Sessa speaks of "detuning of structures" which means reorganising specific system structures into more general ones which are always available; and "diffusion of functionality" which means breaking down constructs so that, instead of having one complex construct to achieve a particular goal, several more basic ones (which can be re-used more easily) should be used. Together these two heuristics are aimed at improving the simplicity of a system by reducing the variety of specialised (context dependent), compound commands.

Metrics which may prove successful in determining if simplicity has been improved deal with such aspects as the number of legal commands, the length of command strings, number of syntactic rules required by the user to issue the appropriate commands, and so on (Reisner 1981; Payne 1984). Assessing simplicity of a UI
requires the analyst to build some model of the system which captures the amount of knowledge required by the system user, in order to complete a given set of target tasks.

Simplicity can be specified as both a formal property (when some syntactic specification language can capture it) and a psychological property of a UI (when it is clear that formal complexity is of the type which users have already represented or can represent easily). However, formal and psychological simplicity are two different things and an HCI technique has to be explicit about whether it has a formal or a psychological view of simplicity. Reisner's BNF Action Language (Reisner 1981) is an HCI UI modelling technique which has a formal view and thus loses out on some of the ways in which humans are able to economise on representation of rules (largely in a heuristic fashion). On the other hand Task Action Grammar (Payne & Green, 1986) uses devices to capture human knowledge representation characteristics which compromise the formal power of the grammar whilst augmenting its predictive accuracy for human behaviour.

2. Compatibility

Compatibility means that the labels (terms or symbols), and the syntax of the interaction grammar do not conflict with users' previous experience. Compatibility and "external consistency" (i.e. consistency with the real world, or some other device; Grudin 1989) are two terms used to mean the same thing; the term "compatibility" is used here as it is less confusing than having several different meanings for consistency. That is to say that the names and grammatical rules that users have already internalised do not interfere with the user's learning, and application, of the rules required to interact with the system (Barnard 1985). The domain of possible real world knowledge to which any principles in this class must appeal has proved to be impossible to characterise in the formal sense. Representations built up by users are difficult to interrogate and are bound to vary, based upon individual experience. Such representations can only be approximated to on the basis of empirical observation and analysis, they are not formally specifiable because they depend upon what exists in the user's head.

Dealing with compatibility means that the systems analyst will have to carry out, and/or refer to results from, research into the nature of the user population and its
background. A combination of human information-processing models and the results of empirical psychological, linguistic and applied research, together with analysis of system characteristics, are all required to come to some understanding of what might be compatible for a particular group of users in any given situation.

3. User-centred task-dynamics (UCTDs)

UCTDs is a term used by Wilson et al (1988) to refer to the way in which UI behaviour relates to the user's representation of task goals and sequences of sub-tasks which the user will apply in attempting to achieve goals. This is another kind of compatibility which operates at a more coarse or general grain of detail than the syntactic/linguistic compatibility defined above. As such the existence of UCTDs also cannot be formally proved because it depends upon what task representations exist in the human head.

Principles in this category are any which refer to task structures represented by users perhaps by appealing to use of analogy (Carroll & Mack 1985) or to psychological models of the representation of plans (Schank & Abelson, 1977; Rumelhart & Norman, 1978) as a means of predicting usability. For the purposes of both design and evaluation, the systems analyst will require information from task analyses (based upon system independent and system dependent task execution) to model the characteristics of users' tasks, with respect to system interaction.

4. Observability

This term refers to the visibility of the virtual behaviour of the system with which the user interacts. Observability, otherwise defined as predictability (Monk & Dix 1987), is formally characterisable, but its validity may be strongly contested if it is not used in tandem with a good understanding of human perceptual and information processing characteristics. It is used here to describe how reliably a system's states and behaviour are represented to the user. If the system provides the user with the necessary information to make decisions which depend upon certain conditions, then the user is less likely to have misconceptions about both whether those conditions are true and about what the consequences of an action will be.

Principles which deal with the issues of feedback, association in learning and so
forth are included in this class. In designing a UI which embodies principles of observability, the analyst will have to adopt an approach which ensures that each aspect of virtual machine behaviour has a UI counterpart which permits discrimination where necessary and which is appropriate to the underlying action. In evaluation, the analyst must discern whether users will be able to build up an adequate representation of system states in order to carry out their tasks. To ensure that this principle is embodied in a system, the designer must evaluate an independent behavioural model of the device, as opposed to a model of the user's device representation.

5. Consistency

Consistency is a term which is frequently used to mean different things in HCI, such as compatible with expectations, or the use of similar structures for similar command expressions, whereas its meaning in Systems Engineering (SE) approaches is more constant and precise. In SE consistency refers to the degree to which operations always have the same effect, regardless of what state the system happens to be in when they are executed (Monk & Dix 1987). In this thesis the SE definition will be used, since the various concepts embodied in the HCI definitions (e.g. Payne & Green 1986, Grudin 1989) are captured by the other types of principle described here.

The class of principles included under consistency covers the avoidance of modality and redundancy in systems design, and the analysis of the degree of existing (psychological) complexity in evaluation. It is difficult to capture consistency without delving into the formal aspects of system specification which can be extremely time consuming and difficult to demonstrate. As with observability, an independent device model (preferably of a mathematically formal nature) is required to capture this property of the UI.

6. Retrievability

Retrievability is used here to mean the ability of the system to permit the user to undo any previous action and to get from any state of the system to any other, without incurring penalties (i.e. unwanted side effects) on the way. This concept includes the functions of "undoing" and "aborting". These functions are essential
because users should not be forced into troublesome or "fatal" situations simply because they accidentally typed in an inappropriate command at the wrong moment. Structured design and programming and formal (in the mathematical sense) analysis of system states and behaviour are the most reliable ways of ensuring that this kind of principle is adhered to, although they are not easy approaches to take. Yet again, the only way to be sure that this principle is embodied in a system, is to build an independent formal device model for the purposes of evaluation.

The above are considered to be classes of principle because they are not dependent on specific circumstances for their validity. However it is fair to say that as, principles with their various implications, they will interact in particular circumstances, and some may have to be traded off against others (Maclean et al 1985, Gould et al 1987). Even in a satisfactory UI it may appear in some circumstances that a given principle has been ignored. It may have been expedient to sacrifice it in favour of other principles, in other words principles may conflict, or become less important in certain cases.

1.2.3 Compromising Principles

In this thesis it is assumed that principles of usability will hold true regardless of EFs. As Thimbleby (1985) states, principles which cannot be generalised are unlikely to be valid or useful. This is not to say that, in UI design, the pursuit of them all simultaneously is possible and will always yield a better interface. In fact, as we shall see, these principles may make what turn out to be conflicting demands on the character of the interface being developed. For example the principle of simplicity, if pursued to its extreme, would render an interface capable of only one state and one action. Such a system would, in fact, not be usable because other principles such as user-centred task dynamics (the system would not support any feasible task a user might wish to undertake, since the one action would have no effect) would be compromised to the extreme. Therefore, these principles can only be applied within the bounds of practicality, according to the "law" of diminishing returns. When the benefits accrued from the application of one principle are found to be outweighed by the losses in terms of others, then it may be foolish to attempt to continue with it.

A requirement for simplicity (say, for example, when errors are to be avoided at all
costs), compromises UCTDs. Put another way, simpler UIs will permit less flexibility and less functionality to support interaction. An example of such a compromise is described by Gould et al (1987) in a design case history where it was impossible to provide adequate and reliable help for system users, so errors had to be avoided by simplifying the UI at the cost of functionality, (although the system itself would have been perfectly capable of supplying extra functionality).

Consistency and compatibility are also often hard to reconcile. Human information processing and behaviour is heavily context dependent, and this fact is compounded by the fact that skilled, contextually influenced behaviour is automatic (Anderson, 1983; Fodor, 1983). This means that, for the system user, a name, or a sequence of behaviour, may have one or more different representations depending upon the context within which the user encounters or elicits it. If the user is behaving in a skilled (automatic) way, he or she may not always behave appropriately within a particular context. Skilled behaviour can be triggered by contextual cues as a result of negative transfer; a concept which refers to the undesirable generalisation of learning. This is particularly likely if that context requires a new representation, or if it closely resembles the context that relates to the inappropriate one (Anderson 1983).

If the system is to be compatible with a user’s representations, then it, like the user, must have multiple, context-dependent representations; it may, as a result, become inconsistent in the Software Engineering sense. For example, the command "move" might be used in a text editor to move text to another part of the screen, but the same command in the "shell" might be used to change the name of a file. The user might see these two as similar operations, although to the system they are very different. This problem would be less troublesome were it not for the fact that users vary tremendously in the way they represent things. It is often extremely difficult to decide whether to sacrifice consistency or compatibility in the design of the UI. Whatever the decision is, there will always be some users who confuse the meaning of names or sequences of actions. The designer has to weigh up the consequences of his or her decisions in the light of such factors as the cost of errors and the likelihood of users being able to learn the correct meanings of things.

Often principles interact in a positive way, for example; achieving consistency makes a user interface more simple (there are less rules for the user to learn when a system is consistent), and observability enhances compatibility and UCTDs
because it increases the available contextual information upon which users can base their representations and plans (Hammond et al., 1983).

1.3 HCI Models

There are many different approaches to achieving usability including application of relevant empirical findings (relating to such things as character size, colour contrast, selection speeds etc) which can provide fragmentory input to design, also use of the specialist's personal experience (not easily communicated, and therefore not likely to advance the discipline of HCI). These approaches may prove useful in applied practice for familiar situations where the physical properties of the interface are focused upon, and where concrete experience has shown which solutions work. However such approaches leave a great deal to be explained in terms of what makes a system usable, beyond physical properties such as character size.

Over the past ten years or so the discipline of HCI has seen the development of a number of task analysis notations, descriptive interaction grammars, and models of the human information processor as tools to improve the design and evaluation of system interfaces. These techniques are all models of the behaviour of the UI and/or the user, which enable us to study interaction without having to contend with all of the problems of variability of the real behaviour itself. Payne and Green (1986), for example use a simple, generative competence grammar to represent the system rules which reduces the problems of analysing real users' representational complexity considerably.

As simplifications of the properties of user- and system-behaviour, HCI and related models are all incomplete, in that they only attempt to capture a small sub-set of the EFs and the usability principles which may determine the character of the interaction between user and system.

In the following, some distinctions are made between types of model on the basis of what approach they take and what aspects of the user or system they focus on.
1.3.1 HCI Design and Evaluative Techniques

The improvement of usability has been approached by the development of two types of UI description. The first, and far less common, type of description involves the generation of an abstract specification of the UI during design. The second is applied to the evaluation of an existing UI specification with respect to the extent to which it embodies certain usability principles.

The arguments relating to the relative values of altering or improving system design practice versus applying usability evaluation methods as means of improving usability have been fuelled by the fact that each approach has its inherent weaknesses as well as strengths. Design approaches are by nature prescriptive and may conflict with the individual needs of a particular designer on a particular project. CLG (Moran, 1981), which is the most influential, complete HCI dialogue design specification method, recommends a top-down specification of the complete dialogue. The levels of specification map onto successive stages in the design process and each level is a description of the whole system, each being more detailed and less abstract than the preceding one. Unfortunately, it is never possible to be sure that early decisions are correct, and the greatest care has to be taken to ensure that invalid assumptions are not made. So early assumptions driving top-down complete system specifications may be misplaced and, because earlier decisions have more profound impact, they can lead to a later requirement for complete re-design which is extremely expensive (Gould & Lewis 1985; Gould et al, 1987; Mantei & Teory 1988).

The HCI alternative is to evaluate specifications or running software of an early prototype, if possible. Descriptions of specified or existing UI software are analysed for their embodiment of usability principles, and qualitative (e.g. Payne 1984, Payne & Green 1986) or quantitative (e.g. Kieras & Polson 1985) predictions about learnability, errors, and other aspects of user performance can be made. The inherent weakness of this type of approach is that by the time specifications of the dialogue structure have been made, or UI software has been written, the design of the system has already become fixed, to a certain extent. The more clearly it has been specified, the less likely it is that the system can be radically changed, since the time and effort needed to make changes are expensive. Furthermore, the earlier an evaluation is made, the less reliable its predictions may be, so an irreconcilable difficulty
could exist.

In the following discussion some distinctions are made between the different focuses of the various models employed by HCI approaches. In Chapter 2 the approaches and their models will be classified according to what they aim to address and the theoretical issues associated with them. In Chapter 3 further issues will be raised relating to their applicability in real (non-research situations) design practice.

1.3.2 Psychological and Interaction models

There are a number of psychological models of human information processing which are not strictly related to HCI, but which are very relevant to the study of usability (e.g. Anderson, 1982, 1983; Barnard 1985; Holland, 1986, and others). These models attempt to capture human knowledge representation and/or processing, learning and association and enable predictions to be made about the ways in which processing will affect observable behaviour.

Interaction models are directly concerned with the ways in which humans behave when engaged in interaction with computers. HCI approaches employing such models may only appeal to psychological properties of users in an implicit manner (e.g. Reisner, 1981; Payne & Green 1986).

Other HCI approaches employ models of both cognition and interactive behaviour (e.g. Card et al 1983, Kieras & Polson 1985) with which they can predict performance of users engaged in such behaviour. The CCT approach of Kieras & Polson (1985), for example, qualifies as both a psychological and an interaction modelling technique because it applies cognitive mechanisms to simulate the control of the user representations of the device. These representations are based upon observed interactions with a UI which are then mapped onto a model of the device in a simulation of interaction. There is really no clear cut distinction between a psychological and an interaction model. The degree to which a model can be said to belong to either category depends upon its purpose or scope, and the strength of its theoretical and empirical bases.

Psychological models can exist at a number of levels of detail. They may deal with high levels of representation such as natural categories (Rosch 1973), the use of
metaphor (e.g. Holyoak, 1984) and so on, or they may deal with low level, neurologically plausible processing behaviour (which has the emergent properties of recognition, recall, association and so on (Anderson, 1982, 1983; Pylyshyn, 1980, 1985; Kosslyn, 1977, 1985), and they may be more multi-level in character (Barnard 1985).

In the same way, models of interaction deal with different grains of detail or levels of analysis (Moran, 1978). The need for different levels of description arises because low level models become too complicated when attempting to demonstrate emergent, or high level properties of behaviour, in much the same way that a quantum mechanics model of the beating of a heart would be.

Psychological and interaction models tend to vary in both scope and grain of analysis. There is no single model that has been successfully applied to explain all of the aspects of HCI (Barnard 1986), even dealing strictly in terms of human information processing, let alone considering social, emotional and motivational factors influencing cognition. This problem is identical to that observed in Information Systems development methodologies where "no one methodology is likely to fit all their [organisations requiring new computer systems] current needs or always be the preferred approach." (Earl 1987).

1.3.3 Competence and Performance Models

We can call the rules that users need to represent, in order to construct legal command sentences for a computer system, the grammar of the language of interaction between the user and computer. These rules may be more or less complex and can be analysed with respect to usability principles.

Similar analysis of structure has long been carried out in the field of linguistics; Lees (1957) defined a grammar as being "a maximally general set of statements which accounts for not only utterances in the corpus [of speech examined by the linguist], but all possible utterances ... Grammar must generate all and only the grammatical sentences of the language".

The rules which a user represents regarding the grammar of the interaction language may be incomplete due to inexperience (Hammond et al, 1982), or incorrect due to
inappropriate generalisation (Moran, 1983, Carroll & Mack, 1985). So studying communication between humans and computers requires that we understand not only the words and what they denote for the user, but also how the user represents the grammar of the communication language.

Both competence and performance models actually describe the structure of communication between the user and system, but competence models aim only to capture a (parsimonious) set of rules that are sufficient to describe the communication. Performance models may apply such rules to represent legal interaction sequences between a user and a UI (usually directed at achieving representative target task goals) using empirically based predictions of unit-action times or error rates as a gross estimate, and possibly applying cognitive processing constraints.

Chomsky (1957, 1965) provides a clear definition of performance models stating that generative (competence) grammars are atemporal, adynamic characterisations of the ideal speaker's knowledge which cannot be evaluated against performance data. Ideal speakers and hearers are defined in such a way that their performance is ignored. On the other hand performance models treat the communicator as a processor, with dynamic properties and constraints, which actively applies the grammatical rules in behaviour.

A competence model of a grammar is only one idealised set of many possible rules which might adequately generate the correct sentences in a language. Acceptance Grammars may be generated by adding alternative rules to a competence model (Payne & Green 1986). A full acceptance grammar would contain all the permissible rules which might be applied successfully to accomplish target tasks. In doing so, such a grammar would be able to capture possible variations between different users.

Competence and Performance models for different systems can be compared with respect to complexity in evaluation. Greater numbers of rules and longer command sentences indicate greater complexity for the human learning the language. On this basis it is safe to assume that with increases in complexity there will be concomitant increases in learning times and error rates.

Also with an explicit specification of the rules required to interact with each system
it will be possible to predict problems for users of one system moving onto another. Use of grammatical models of the respective interaction languages can enable the analyst to identify mismatches between the sets of rules which would indicate sources of difficulty and confusion for users due to negative transfer of knowledge from one system to another.

It should be noted that the kinds of grammars used in HCI competence and performance models are largely context free (Fountain & Norman, 1985). This means that they do not easily (if at all) capture the influence that context has upon the representation of rules. Furthermore they lack scope in terms of the human behaviour which they are able to address; for example perceptual processing and semantics are given little attention by Card et al (1983) and Kieras & Polson (1985). For this reason, usability principles such as compatibility and observability are difficult to address using these techniques.

1.4 Questions of Appropriateness and Applicability for HCI Approaches

In the following sections, considerations which may have to be born in mind by an HCI analytic approach are illustrated. Regardless of its nature, be it a design or an evaluative approach, incorporating a psychological or an interaction model addressing competence or performance issues, a global view of the context of the design or evaluation must be maintained.

The above discussions serve to illustrate that there are many different perspectives on and approaches to the design and evaluation of usable UIs to choose from. These include the EFs which are taken into account for the purposes of judging usability, the suitability of improved design techniques versus rigorous evaluation techniques, application of more complete, psychologically valid human information processing models versus simpler, pragmatic models of processing and representation, and competence versus performance models which differ in many ways such as the nature and value of predictions yielded, and prediction of user’s learning difficulty and errors. There are so many factors and approaches, which could influence ease of use, that the task of choosing and applying an appropriate approach to design and evaluation, which will highlight the important properties of a UI, is daunting to say the least. The designer may believe that any particular approach will take time to
apply properly and that there is only enough time for one. How might a designer choose one of the approaches?

No single approach yields all the answers, but applying even one of the above types of methodology will very probably require a considerable amount of time and care as will be demonstrated by the approaches discussed in the following chapter. These approaches require the analyst to devote a great deal of effort to analysing tasks, specifying task structures, or system and UI behaviour and possibly modelling the user's representations of any of these things. All of these activities will naturally take up valuable time, so it is unlikely that systems designers will have enough time and resources to ensure that every aspect of usability is adequately dealt with. In other words it would seem that a question of compromise is at stake whereby it is necessary to determine what aspects of a UI deserve the most attention in a given UI design.

In order to assess what aspects of a UI might be most important for the design in hand, the designer will need to determine what features of the system and its users might play a major role in constraining the options available. A prioritisation exercise may well be necessary. It is true to say that different computer systems have different demands placed upon them, and require different types of UI. What principles of usability turn out to be most important for a given interface may vary depending upon the nature of the EFs defined earlier, for example if users are highly trained, then compatibility of command names may not be as important as UCTDs which ensure maximum efficiency in task execution. These EFs will be crucial in determining the parsimony of adopting any particular design or evaluation approach as opposed to others which might be available (for example, an approach which does not account for the behaviour of the application, may not be capable of showing whether the user's representation of tasks is well supported; Kieras and Polson (1985) have shown that representation of device behaviour, as well as user task representations, can enable a simulation to predict areas of task difficulty).

The EFs introduced towards the beginning of this chapter are elaborated here, rather than earlier because they impact upon the interpretation of the importance of usability principles described later on. Having considered both usability principles and different types of HCI approach which attempt to embody them it is useful to
reconsider the meaning of the different factors with respect to possible design issues. These factors have to be considered by systems designers, and any HCI approach which they might use should attempt to support them in their efforts to do so. A number of influential approaches which might be candidates for such purposes are described in the following chapter.

Recalling the earlier assumption:

**Usability may be said to be a system feature when:**

\[ U(s) \times S_{\text{App}} \times UI \rightarrow Acc(TTs) \]

These EFs essentially determine the value of any design solution and the type of analysis applied to the usability of a UI. In the following subsections, some examples of how EFs impact upon both design (because all design practice must be exposed to evaluation itself) and evaluation of UIs are discussed.

### 1.4.1 The system user(s);

**An Individual or a Population** \( U(s) \)

It is usually the case that systems are designed to be used by a population of users. Different people can have vastly different requirements from a system, this is because they are all *individuals*. They may vary in the kinds of tasks they want to do; some users may only use the system for one type of task, others for a wide range of tasks. They may differ in their experience with similar or different applications and with other UIs, consequently compatibility and UCTDs are principles which require empirical investigation if they are to be satisfied. Users will very probably vary in the length and frequency of their sessions with the system and the amount of training they will have had. They will also have different attitudes to (and aptitudes for) use of the system. In these and many other ways users prove to be unpredictable. Furthermore, in some cases the variability of users will be greater than in others. The system designer will have to build an extremely sophisticated system if everyone is to be satisfied. Otherwise some users' requirements will have to be compromised for those of others, and this is almost always what will in fact happen.

The best design solution is likely to be one which satisfies most of its users most of
the time. Making a system so simple for the sake of naive users that experts have to
go to a great deal of trouble to carry out more complex tasks may not be acceptable.
Consequently naive users may have to devote some effort to learning a system that
has been designed more explicitly to support relatively expert users with more
powerful commands. The more variable the user population, the more difficult it is
to come to a simple compromise.

Some examples of the issues that the UI designer will have to address with respect
to a variety of users will include:

Experience and expectations.
Training.
Learning opportunities and forgetfulness.
Attitudes and social aspects.

1.4.2 The Nature of the System Application(s) SApp

This is taken to mean the structures and behaviour which the virtual system mani-
fests once implemented, being a function of hardware, and software.

A complete computer system may incorporate multiple integrated applications (an
application is essentially a package of software and the jobs it does with and for a
user. It may be a whole system in itself, or merely a sub-system such as a drawing
tool or a database) and these and their UIs vary immensely along with the uses to
which they are put. Applications are generally classified by systems analysts into
types, the largest two being data processing and real-time. Data processing applica-
tions are largely concerned with the manipulation and transformation of data which
may relate to structure and operation of an organisation, records kept on business
transactions, stock control, and so on. Real-time applications are those which have
to respond within tolerance limits to external events, such as process-control sys-
tems, navigation systems and so on. In these cases there are external inputs to the
behaviour which the virtual machine exhibits, and users are expected to direct the
system to interact with some aspect(s) of the external environment. There are also
other types of application such as the paper substituting text-editing systems and
office systems which are used to store and organise the kind of information which
otherwise fills filing cabinets, diaries, in- and out-trays etc.
The functionality of the system is of course strongly influenced by its application(s). The behaviour and performance of the UI for a given system has to be evaluated in terms of its suitability for the many kinds of tasks which it is meant to enable the user to execute. Its success in so doing can only be determined with respect to the system functionality within each of its applications. There may be little to be gained from criticising the UI for failing to support tasks which the application would be incapable of handling.

The different functional requirements of applications mean that UIs will be also subject to different design constraints and performance requirements. Very different error tolerances are allowed for text-editing applications and for air traffic control systems. It may be acceptable for a manager to delete a file accidentally from time to time, but it is not acceptable for an ATC operator to fail to notice an impending head-on air collision. So the UI must be designed to ensure that the application is appropriately represented to enable users to complete their tasks with acceptable reliability. In order to do so the following issues amongst others have to be considered:

* System Scope (the extent of allocation of functionality to the system as opposed to the user).
* Nature of functions the system must accomplish.
* Nature of the user's job.
* Reliability of user-system performance (with respect to risks and performance requirements).
* Feedback (when necessary).

### 1.4.3 The System Interface UI

This is defined as the type or style of UI or its various characteristics.

The previous two EFs (U$k$ and SApp) discussed partially pre-determine some of the necessary features of the UI however, there is still a requirement for designers to choose between certain options. Such a choice may depend upon the sophistication of the available, and affordable, system hardware and software with which the UI designer is working. Familiar types of interface currently available include; command-driven; menu-driven; window based; direct-manipulation; WYSIWYG;
and others. Each type of UI has a different impact on what is possible in terms of functionality for the user, and with the possibilities come new design problems. For example, once the user has been provided with the opportunity to run several processes on a system simultaneously, the designer must consider what the usability implications might be such as forgetting to terminate background processes or confusing one environment with another. Issues which might arise during design and evaluation are:

*Forms of information presentation, and how much information.*
*Functionality of the available tools, packages.*
*Potentially hazardous or difficult states users can get into.*
*Additional functionality such as help to enhance usability.*

1.4.4 Target Task(s) TTs

This is defined as; a set of target tasks which the system must support.

Systems, whatever their user population, applications or type, are normally designed to support some agreed minimum set of target tasks; this may determine the scope of the system application. The scope of the system application in design provides a specification of the extent of its functionality. However, the users’ target tasks themselves are a more precise definition of the system functionality than the scope because they will impose more structure, in terms of sequence and interdependency between operations. For example, the scope of a system may include both database management and editing functionality, but only an explicit specification of the user’s target tasks will clarify the fact that it may or may not be absolutely necessary for the two.

The target tasks may be explicitly stated, in some business contract between a design team and a client, or they may be a more informal collection, perhaps with a view to expanding the set as far as possible. The difficulty in deciding whether to support any given task, or set of tasks, is in deciding what is necessary, what is only desirable, and what is actually counterproductive. In the study referred to earlier by Gould et al (1987), an example of the design of a messaging system was given where certain desirable tasks, which users might have wished to carry out, would have rendered the system more error prone.
In many circumstances there may be a conflict of interests between users, purchasers (e.g. client management) and designers as to what tasks ought to be supported. Users want a system to support whatever they want to do with it. Managers want the system to increase productivity. Designers want their job of producing an efficient system, which successfully supports users' requirements, to be as easy as possible (which it will not be if substantial requirements drift occurs, as it often does, particularly in large projects (Appleton 1986). Examples of issues which may be important with respect to decisions about target tasks, which the designer may have to account for, include:

Validity of set of to-be-supported tasks specified by clients/users.
Necessity, desirability and counterproductivity of tasks.
Psychological and organisational implications of target tasks.
Feasibility of system support for tasks.
Allocation of functionality (requirements for system support).
Likelihood of task errors (the user carries out the wrong, but legal step, as opposed to illegal user actions)

1.4.5 Acceptable Interactive Performance Acc

This is defined as performance matching or exceeding some performance criterion or criteria.

All systems have associated likelihoods of failure, either in their functionality or in their interaction with users. When the interface misleads a user, or permits the user to make a mistake, it is often the user that is blamed for making that error. It is true that humans can be negligent, but even if they are not, errors sometimes occur. In other words interactive performance can hardly ever be expected to be ideal, and ideal performance can never be guaranteed. Clients for systems often stipulate performance requirements for a system in terms of what is acceptable in terms of some measurable criterion such as speed and/or reliability. The minimum requirements may depend upon a number of issues the most important being costs of the consequences of failure. When lives, or very high costs are at stake, then significant risks of errors occurring are not to be accepted. However when the consequences of poor performance are not great, then other considerations, such as speed, may take priority, and high error rates may be tolerated.
The client and the designer may negotiate over the performance criteria, especially if the client makes demands which require a more expensive project or product than the client can afford. The UI designer should consider, amongst other things:

*Costs of meeting performance targets.*
*Necessary trade-offs in system functionality.*
*Evaluation of interactive performance.*
*Reliability of evaluations.*

### 1.5 Summary

The importance of maintaining a broad scope of perspectives in system design and evaluation cannot be over-emphasised. Researchers in HCI are beginning to accept the importance of breadth as a necessary quality of approaches which can be expected to produce realistic assessments of the usability of UIs (e.g. Barnard, 1986; Green et al, 1987; Grudin, 1989). The preceding sub-sections attempt to illustrate the diversity of considerations which impact upon the usability of a system. Five evaluation factors (EFs) were introduced which aim to provide a view of the breadth of important issues which exist in the context within which interactions take place between users and systems. An crucial question is whether this breadth is apparent in HCI approaches to design and evaluation. If it is not then it raises serious concerns about their applicability to or usefulness for actual design practice.

Chapter 2 aims to examine a number of current techniques in terms of their type, scope and character in some detail. This review is necessary before we can assess the usefulness of each technique or the methods it offers for improving system usability in a real design situation. Amongst other things it is necessary to determine its ability to account for the above EFs, either in the design or in the evaluation of a UI, and what principles of usability it focuses on. In later chapters we will be attempting to determine how well each technique might work in a more practical sense, in order to decide whether it is appropriate for application in a commercial design project. Chapter 2 will therefore be restricted to an analysis of these techniques from a mainly theoretical point of view.

The rationale for addressing the theoretical and the pragmatics of these techniques separately is that, to date, there has been a dearth of research on the use of such
techniques in real design environments. In these situations the nature of the UI being designed may be different from that which a given technique was tested on, the UI designers may have no psychological expertise, and resources may be inadequate. Also the technique will have to integrate with the rest of design activity both in terms of other techniques carried out and in terms of sequence and timing. Since most of the available literature on these techniques is based upon controlled research, little is known about how these issues affect the techniques, and how they may need to be modified for practical use. The aim of this research is to address precisely this question. So issues of application will be raised throughout the discussion following Chapter 2.
Chapter 2

A Review of HCI Design and Evaluative Techniques

2.1 Introduction

2.1.1 Overview

The HCI analytic techniques discussed in this chapter are:

- Block Interaction Models
- Reisner’s Formal Interactive Grammar
- Task Action Grammar
- ACT*
- GOMS
- Cognitive Complexity Theory
- Interacting Cognitive Subsystems
- Command Language Grammar

These analytic techniques are called design and evaluative techniques because they are intended, or can be used to analyse user interface designs in terms of their usability. Some of them are intended more as research tools, but some authors claim direct utility for their techniques in actual design practice. It seems reasonable therefore to take the view that they may be criticised as proposed design and evaluation tools since one or both of these purposes are their main focus.

However, this chapter seeks merely to summarise the main features of these techniques as scientific analytical approaches without tackling further issues concerning their actual applicability to design practice as it occurs in the "real world". The reason for clarifying the approach and scope of each one is to provide a clear basis for consideration of the implications of use of these techniques within applied and commercial design practice. Such considerations are restricted to later chapters.
The eight models (or modelling approaches) reviewed in this chapter are a small selection of some of the most influential HCI descriptive or predictive techniques which have been developed for the purpose of designing or evaluating UIs. Some are pure research tools, others are claimed to have direct value for the design of more usable systems. The techniques under consideration all share the characteristic of the embodiment of some property of a user's knowledge representation or processing properties within a specification or model of the user or grammar for representing dialogues or interaction rules for some UI. They are by no means a complete set, but are representative of an important class of design and evaluative techniques currently available and based on work carried out in the field of HCI.

2.1.2 Classifying and Characterising the Scope of HCI Techniques

The above approaches will be discussed in terms of the structure outlined in the following section. Each model will be classified with respect to whether it is an explicit or potential Design Technique or an Evaluative Technique; whether it models Psychological aspects of the user and/or Interaction between the user and system; and whether it is a Competence model or a Performance model. The scope of each model (both in terms of what the model aims to address and in terms of how it does so) will be described along with the devices used to capture the various aspects of interactive behaviour. The models will also be characterised according to the evaluation factors and usability principles they address. This characterisation is essentially a concise means of representing the aspects of a UI and the evaluation factors addressed by each model. The characterisation takes the form of a scoping matrix with a number of possible cells which the model may tackle. This permits a clearer comparison between the different models.

The Usability Scoping Matrix in table 2.1 illustrates all of the ways in which classes of the usability principle and evaluation factors described in the previous chapter can intersect. As stated above this matrix attempts to characterise the possible scope of any HCI Design or Evaluative technique in terms of what aspects of usability and what evaluation factors it captures, however it makes no statement about the methods and devices involved in each model. In the following discussion, each principle (represented as a column of cells) will be discussed with reference to how the evaluation factors might impinge upon it's analysis. The last row of cells (the intersection of each usability principle with acceptability of performance) is the
most critical for any model which seeks to influence system design. Models which
do not address the qualitative and/or quantitative performance aspects of HCI can
only be considered as research tools which perhaps further our understanding of the
relevant questions which need to be addressed by further work in the field.

A Brief overview of the implications of table 2.1 follows.

Models which capture:

(a) **Simplicity** must model (and present metrics for) complexity of representations
or specifications.

(b) **Compatibility** must capture the relationships between external (real world, and
other systems) knowledge and the knowledge required to use the system.

(c) **UCTDs** must present system or device independent task analyses and goal struc-
tures which reflect the needs and characteristics of the user.

(d) (e) & (f) **Observability, Consistency and Retrievability** must present user in-
dependent representations of the appropriate system properties.

Models which account for:

(g) **Users** must employ empirically based psychological theory or preferably expli-
cit models of human information processing.

(h) **System Applications** must entail representations of system functionality and
behaviour.

(i) **UI Type** must be sensitive to static and dynamic representation and behaviour in
interaction (e.g. command versus menu driven).

(j) **Target Tasks** must specify a coherent (or perhaps representative) set of basic or
composite tasks the system is designed to support.

(k) **Acceptability of Performance** must generate testable predictions about qualita-
tive or quantitative aspects of user behaviour.
The way in which the principles of usability might be addressed in terms of each of the evaluation factors is elaborated in the following discussion. Approaches must meet certain requirements if they are to succeed in addressing each cell in the Usability Scoping Matrix.

2.1.3 Usability Principles With Respect to Evaluation Factors

The Principle of Simplicity with respect to:

Users

Simplicity as defined in Chapter 1 can be described and evaluated in terms of the complexity of the rules required in models of the representations which users might build of a system. Such a model may be more or less realistic depending upon the extent to which it accounts for users' existing knowledge structures and cognitive strengths and weaknesses. Real users representations may or may not reflect the reality of virtual system structure and states, depending upon how the system presents itself, and upon their previous experience. The approach covering this cell must embody a psychologically plausible, empirically supported model of the user which can capture complexity.

System Application

Some applications are, by necessity, more complex than others regardless of the nature of the UI. Additional complexity can be caused by poor UI design, but each application imposes a limit to how much improvement of UI simplicity can result in a system which is overall simple to use. The more functionality a system has, the more rules a psychologically valid UI model which attempts to capture simplicity may be expected to require. However this complexity may not be sufficient grounds for dismissing a system with greater functionality. Therefore an approach represented in this cell must capture necessary UI behaviour (that which is determined by the nature of the application’s behaviour), as distinct from interactive UI behaviour (that which serves merely to mediate between the user and the application). To do this it must have an objective device representation.
## Table 2.1
Scoping Matrix of Classes of Usability Principle Over Evaluation Factors

<table>
<thead>
<tr>
<th>Evaluation Factors</th>
<th>Usability Principles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simplicity</td>
</tr>
<tr>
<td>Users (Us)</td>
<td></td>
</tr>
<tr>
<td>System Application (App)</td>
<td></td>
</tr>
<tr>
<td>User Interface (UI)</td>
<td></td>
</tr>
<tr>
<td>Target Tasks (TTs)</td>
<td></td>
</tr>
<tr>
<td>Acceptability of Performance (Acc)</td>
<td></td>
</tr>
</tbody>
</table>
The UI itself may impose complexity on an HCI model. UIs and their dialogues may be designed in such a way that each representation and command sentence is differently structured and there is no coherent scheme or grammar which can enable the user to capture generalities. Consequently more rules are required to interact with the system.

The UI can also impose complexity through its style. Direct manipulation menu based systems take the onus of memory retrieval of available options and entities away from the user who simply has to recognise the required command name in a menu or icon on the screen. The approach must be sensitive to differences in operation of, behaviour of and presentational characteristics of the device.

Target Tasks

The target task set may be large or small, and the actual tasks and sub-tasks a user may be expected to be able to carry out using the system may themselves vary in complexity. For example, an editing system may only be used for very simple tasks such as typing in text and basic editing operations such as DELETE, INSERT, COPY, MOVE, and so on, or it might be used for more complex tasks such as "search for a string and substitute with another at every occurrence", or "draw a figure with labels". The approach must be able to capture the complexity of task structures.

Acceptability of Performance

Whether a model is capable of addressing performance, as it is affected by the simplicity of a UI, depends upon how the above cells in the simplicity column are addressed. It may be possible for a model to produce predictions without addressing all of the above, however it may be unwise to base assessments of simplicity on a UI description alone, for example, since one then has no knowledge of the characteristics of the prospective users who may or may not have all of the necessary knowledge required to operate the system in question.

A technique which addresses the first four cells in this column may still not fully ad-
dress the last cell if it makes no quantitative or qualitative performance predictions such as increased errors or speed of task completion which can be compared with performance criteria which are judged acceptable by some analyst or commercial client of a system design team. **The approach must make testable predictions which would be suitable for assessing acceptability of user performance**

The Principle of Compatibility with respect to:

**Users**

In some senses compatibility is strongly related to the simplicity-users cell in the usability scoping matrix. Compatibility of a system may make it much easier to learn because fewer new rules need to be acquired by the user. User compatibility of a UI is one of the most difficult of all of the usability principles to characterise because it depends on the huge domain of all knowledge which the user brings to the system. This knowledge is highly variable and involves semantic, as well as procedural knowledge (Anderson 1983). **The approach must be able to map UI behaviour and semantics onto users’ processing and representations and show mismatches between the two.**

**System Application**

Modelling UI compatibility with the system application dependent UI features may be a more simple affair because systems are easier to model than users are. Compatibility between the objects, dialogue and behaviour of the UI and the objects and behaviour of the application (where the behaviour of the one is reflected in the behaviour of the other, also termed congruence and semantic-syntactic alignment by Payne & Green (1986)) relies on mapping common terms, operations and behaviour from the actual context or the traditional domain of the application, to the UI. For example, it is a simple affair to represent temperature using height in a graph (this maps well onto traditional temperature representations in thermometers), and icons representing the application’s pre-computerisation devices are commonly exploited (e.g. file holders, bins etc). **The approach must show that system application behaviour maps well or poorly onto UI representations**
UI

The nature of the UI may heavily influence the degree to which compatibility can be achieved, so evaluations have to take into account the constraints or advantages presented by the system characteristics. For example, the system graphics may be inadequate to represent familiar (compatible) icons clearly, input devices may be poorly suited to the kinds of manipulative skills users already possess, multi-tasking facilities may or may not exploit the human capacity to work in a heavily context dependent fashion (as demonstrated in much of the literature on processing, representation and metaphor; Anderson (1982), Anderson and Ortony (1975), Carroll and Mack (1985), Holyoak (1984). The approach must be sensitive to operational and perceptual system differences and be able to show incompatibilities with user characteristics.

Target Tasks

Compatibility of target tasks is in fact a usability principle in its own right. So the third column in the Usability Scoping Matrix is really an expansion of this cell, and any model which tackles intersections in the UCTDs column will be represented in this one. The issue of compatibility of target tasks may be best addressed by focusing on the goal structures users have, and possibly identifying those structures which are a result of the constraints imposed by the traditional practices of the domain and those which have a more fundamental root in the logical order in which the stages of a task must proceed.

Unfortunately there is a dearth of research on this kind of task analysis. The research is usually controlled in such a way that users are trained to perform tasks according to an imposed goal structure, perhaps from a manual (e.g. Kiers & Polson, 1985), or it is assumed that users goal structures can be derived from the constraints imposed by the traditional domain or by alternative applications expressed as perhaps a set of external task structures (e.g. Moran 1983).

The approach must be capable of mapping the target tasks' goals and structures onto users' representations of the target tasks, in order to pinpoint areas of mismatch.
Acceptability of Performance

Predicting the acceptability of performance may be heavily tempered by an understanding of the compatibility issues arising in design. This would require a model of the semantic as well as procedural or syntactic aspects of user processing, perhaps based upon a task analysis or other empirical research. Understanding representation and meaning for users and mapping it to representation and meaning in the UI would facilitate identification of incompatibility between the two representations.

An approach which has the power to model performance differences between recognition tasks, skilled activities, novel problem solving or decision making and so on will have the ability to generate performance predictions with respect to compatibility versus incompatibility of UI designs. The approach must generate testable performance predictions based upon degree of, or differences in, compatibility.

The Principle of User Centred Task Dynamics (UCTDs) with respect to:

Users

When considering UCTDs it is of course the users and target tasks which are the focus. Goals and methods in the UI, designed for the system-supported task executions, must be modelled. So must users' goals and task representations on the basis of empirically or theoretically derived information about how users process, represent and employ task related knowledge. The approach must be capable of capturing possible users’ task representations and in such a way that it can show mismatches between users' actual and ideal task representations for use of an application (i.e. required for error free performance) which should indicate sources of difficulty in interaction.

System Application

Again, the analysis is tempered by considerations of the constraints imposed by the system application which may be inherently problematic, calling for complex manipulations of data, or parallel activities (at which humans are notoriously poor).
Some of these may be related more to the system application than to inherent characteristics of users and tasks. As with compatibility, it is important to avoid confusion between system application-independent and dependent factors. The approach must be able to distinguish system application dependent task structures and show how these do or do not map onto users' representations of task structures.

UI

There are also the limitations of the UI design (independent of application dependent enforced task structures) which may be poorly suited to the kinds of task activity necessary. Consider the problems of displaying large chemical-plant systems on a relatively small computer screen with poor resolution (even with two dimensional scrolling facilities) as opposed to a large hard wired display mounted on a wall. The task of locating a blocked valve on the computer version of the plant representation in response to an alarm would be much more complicated than with the hard wired version, involving identifying and calling up the appropriate screen, as opposed to looking for a flashing light on the wall. The approach must be sensitive to and show how UI task representations might be well or poorly suited to user task structures.

Acceptability of Performance

The ability to predict acceptability of performance will require a model to predict how tasks will be accomplished on the system given the difficulties of the application, the limitations of the UI and the way in which users will represent knowledge required to carry out tasks. The approach will have to produce testable performance predictions with respect to task structures.

The Principle of Observability with respect to:

Users

Representation and feedback are two of the important issues which arise in UI design. To determine whether system users are actually presented with, and successfully perceive, the necessary information to successfully achieve their tasks has
been a major aim of both ergonomics and HCI, and may be a relatively easily characterised aspect of a UI. Having said this, there are, as always, problems. It is not so much the matter of presenting the relevant information when it is required, but how that information is presented which gives rise to questions of usability (there can be too much at once, or it can be in an inappropriate form).

There is a potential dilemma for observability in that it is not always easy to distinguish between necessary information and additional relevant information. The designer will always be aiming to provide enough information about system states and options for the user, without swamping the entire screen with information that a user might need. Perhaps the best way to ensure this is to user-test a prototype of the system, but if this is not possible then a performance model which can represent the user’s information requirements for task execution, together with processing constraints, might be able to predict whether user performance might be acceptable given that certain specified system states and options are observable, and others not. The approach must show that information presented to users is both sufficient for their goal and task requirements and perceived reliably and not misinterpreted.

System Application

If we assume that when it is presented, information is usefully interpreted by the user, then we still have to be sure that the behaviour of the system application, where it is relevant to users’ requirements, is always represented (obviously this implies a need to consider the UI cell for this column also). The approach must show that device states which users need to know about are always matched by unique interface states, or at least that this is the case for all the states a user is likely to get into in normal use.

UI

Observability must be achieved within the limitations of the UI type which can greatly compromise observability, as, for example, anyone who has ever used a non-WYWIWYG text formatting package will know. For example, some UIs may have to represent function selections with labels only whilst others may be capable of presenting icons, or labelled icons. We need to know what is best given the limi-
tations of the UI and whatever style of presentation is adopted, and we can only do so if we can identify the salient aspects of these representations for the user (in other words although we can prove that something is observable and even discriminable in a formal sense, we cannot guarantee that it will be so for users, and we need some constructs which might enable us to address perceivability). The approach must be sensitive to presentation characteristics of the UI type and how these impact upon users’ ability to distinguish one representational option from another.

**Target Tasks**

The analysis must be able to determine if all of the information required to accomplish target tasks is really presented by the UI. If it can be shown that necessary or desirable information is not presented to the user by a UI design, then it is safe to assume that there is a real probability that performance will be compromised. The approach must specify what the states of the system, and the user’s representation are with respect to the task in progress and show that all requisite information is passed to the user by the UI.

**Acceptability of Performance**

In order to determine how performance is affected by presentation, we need to be able to model both perceptual processing, salient (relevant to the user’s tasks) system behaviour, and how this behaviour is represented at the UI. In other words perhaps all the cells in this column are crucial to being able to address performance adequately. The approach must be able to predict some variation in performance characteristics, based upon the presence or degree of clarity or redundancy, and nature of, observable events in response to application behaviour.

The Principle of Consistency with respect to:

**Users**

The simple definition of UI consistency given in Chapter 1 has rather complicated repercussions in HCI modelling. Users do not have consistent representations for labels and procedures. Humans are notoriously context dependent in their processing and interpretation of events they perceive (Fodor 1983), as has been noted, but
the nature of the moderation of behaviour by context is not the same for all users as there tends to be a great deal of individual variation in this respect (for example, at first glance a fly-fisherman might interpret the word "bank" somewhat differently to the way in which a financier would). The approach must be able to show that the system is indeed consistent, but, to appear in this cell, it must also be able to determine where this will conflict with users' representations in order to determine whether one or the other needs to be changed.

System Application

The same is true for system applications as is for users, where the same action can have radically different effects depending upon its state or context. For example, in a real-time, process control system, the act of adding a chemical to a batch process may produce a desired reaction in some circumstances (if the other chemicals in the batch and the temperature are appropriate) or an explosion in others. The approach must be able to determine whether there are features of the application which force inconsistency on the UI. This can then be dealt with by adding information about states which will enable users to distinguish states.

UI

If a UI is not able to represent certain system states then some commands may result in apparently inexplicable inconsistency. For example, if a file has been protected so that it is not alterable, and the UI is one that cannot show a status bar, then user commands that would normally cause the file to be edited, or deleted, would inexplicably result in no action. Another example of UI inconsistency is that of the effect of pressing different mouse buttons in text-editing tool windows or in command windows on a Sun workstation, where each of the three mouse buttons may be used for a different effect. The approach must be able to identify UI characteristics which will force inconsistency in the effects of operations between different system states which may not be perceived as such by the user.

Target Tasks

Certain target tasks may enforce changes to the system application state which will affect the consistency of the effect of carrying out target tasks with a system. For
example if a user has to format discs before using them, and formatting a disc destroys files, if there are any on it, then the effect of the task of formatting a disc is, as far as the user is concerned, inconsistent depending on whether he or she attempts to format a new or an old disc (although the application, and the UI may well be behaving in a completely consistent manner). The approach must identify the effects on the system states of target task actions with respect to users' task goals and determine which tasks will have inconsistent effects depending upon system states.

Acceptability of Performance

Models characterising system consistency may assess acceptability of performance if they can formally describe the UI behavioural properties for all states (Dix et al 1986). Sadly, adding the user and the application to this equation complicates the matter astronomically because users do not perceive consistency in a manner which is regular (all individuals are different) nor are they free from context effects. The approach must at least be capable of determining what the most probable system states will be and address consistency of operations within these. It must be able to predict variations in performance based upon changes in the consistency of the device. However, consistency considered independently of other usability principles may well be a misleading or false indicator of usability (Grudin 1989).

The Principle of Retrievability with respect to:

Users

Assuming users often wish to undo mistakes, or find their way back to a previous state, or to another part of the system, retrievability is an essential property of a usable system. It should be possible to model the kinds of system states users are likely to get into (which may involve moving around a system whilst in the middle of some task such as editing or running programs). Having done so a model might suggest that given likely user behaviour, a system will allow undoing, interrupts and parallel processes to be run to the extent that a user will not be constrained by irretrievability. The approach must be capable of formally proving that at least the states which users are likely to enter can always be safely exited, and that all
parts of the system the user might want to get to are reachable from any state
the user is likely to be in.

System Application

If the system application is such that users are likely to make irreversible changes to
states, for example deletion of files and directories, or addition of some chemical to
a batch process, then this clearly has implications for the degree of retrievability
possible. The approach should be capable of distinguishing reversible from irre-
versible actions, by modelling actual system states, as opposed to task struc-
tures, UI states or user representations

UI

The UI may permit the user to see more clearly the irreversible consequences of
commands or tasks. It may display information which helps the user to find a way
out of a state, or a way of reversing the effects of commands. For example, the ap-
plication type may be such that it is not possible to provide all of the functionality
that would be required to ensure retrievability. The application might involve de-
struction or processing of data or things which cannot be reversed. It may lack func-
tionality such as buffers for storing data which has been deleted, but which might
need to be un-deleted. Designers might wish to respond to such problems by adding
confirmation steps in command sequences at the UI for such actions, thus reducing
the likelihood of users unintentionally entering states with irreversible conse-
quences. The approach appearing in this cell should be able to determine
whether entry into desirable or undesirable system states is made more or less
possible and probable by features peculiar to the UI involved

Target Tasks

Retrievability may be assessed simply by formally describing all of the possible
states a system can be in. However, the size of such a description for a real system
application with one of the more modern graphical types of UI tends to be consider-
able and it is probably more realistic in applied design to prioritise analysis accord-
ing to which actions in which states are more likely as Monk and Dix (1987) have
done, or where the consequences of being in such states are very serious. The ap-
proach must be capable of representing the retrievability of actions required within the target tasks, particularly if they imply irreversible actions and if they have serious consequences.

Acceptability of Performance

Evaluation or prediction of human performance with a UI, with respect to retrievability may depend on a partial analysis driven by the nature of users, the tasks they will need to carry out, the necessity of irreversibility and its seriousness, and so forth. With the introduction of cheap automated systems analysis tools it should become possible to design systems which really are, formally speaking, foolproof without spending an intolerable amount of time (and money) on the exercise, however this seems an unrealistic goal at present and such an effort is likely to be both expensive and time consuming. A combined human performance and system state model would be required to predict the occurrence and costs of errors enabling a designer to choose between constrained relatively human-error proof UIs and more flexible, faster but more human-error prone UIs.

We have now considered examples of what it might mean for an HCI technique to address each of the cells in the usability scoping matrix. In the following subsections each of the HCI techniques discussed is characterised in terms of the matrix which gives an impression of the principles it addresses, and from what perspectives it does so.

2.2 HCI Models

HCI Design and Evaluative Techniques (DETs), are approaches to providing useful, descriptive and possibly predictive models of human-computer interaction (or of some aspects of this). They represent a substantial component in current research efforts, and are aimed at providing some theoretically based approach to the job of design or analysis of more usable UIs. Furthermore, they may act as a guide for practitioners who might be interested in applying a rigorous approach to user-oriented design or evaluation, but who do not have the time or expertise to work from first principles.

The eight techniques reviewed here are varied in their approach and aims. The
thing that they all have in common is their ability to represent and/or predict knowledge required to interact with a computer system or the behaviour of the user or system interface in a form which highlights interesting properties of users or UIs which have some bearing on the usability of a device. The purpose of this review is to provide a flavour of the nature of HCI DETs for the purpose of assessing their impact upon applied and commercial design practice. Where the models are characterised (see each of the usability scoping matrices; tables 2.2 to 2.9) the assumptions about their scope are generally drawn from statements and claims made by the authors rather than from empirical evidence in applied design since such evidence is scarce or non-existant in most cases.

The order in which these models are presented is based upon sophistication and historical dependency. It is therefore appropriate to begin briefly with a general model which attempts to characterise the breadth of user knowledge which might potentially be accounted for by any HCI descriptive approach. The Block Interaction Model (Morton et al, 1979) presents a useful framework for representing the sources of a user's knowledge relating to a particular task or problem.

2.2.1 The Block Interaction Model (BIM); A Model of Models

Classification of the Model

The BIM cannot really be classed as a design or evaluative technique because it suggests little in the way of its practical applications for either of these purposes. It is more descriptive than prescriptive and might be used in either design or evaluation as a way of viewing a space of possible design or evaluation measures which might be appropriate for some system in a particular context. A BIM is not sufficiently detailed to qualify either as a competence or a performance model since it only represents the sources, and not the content of knowledge. It does qualify as a psychological model, rather than as an interaction model, because it is based upon an analysis of what sources of information contribute to a user's representation of a system, and some of the interactions which might occur between these sources.
Scope of the Model

Figure 2.1 shows the generic BIM which describes some of the different kinds of knowledge which might influence a system user’s representation of a current problem or task (for any given example it should be possible to include more specific information about the nature of such knowledge). These knowledge sources interact with one another in such a way that it is not possible to say that a user will represent a given procedure or input in an immediately obvious way.

Users bring knowledge that will enable them to predict system behaviour and guess certain things correctly, but they also bring knowledge that will cause them to make mistakes due to erroneous assumptions or beliefs. What users have is a mixture of interdependent ideal and non-ideal knowledge which they will tend to generalize consciously or otherwise to their interactions with the system. This process of generalization is known as transfer of learning, and it acts in both positive and negative ways on user performance.

In the BIM double bounded boxes connected by double lines are used to represent the primary information sources and interactions between them in an ideal user. The real user also utilises what is represented by the other boxes and lines, and it is these extra knowledge sources which (as well as embodying some useful and helpful information) are the sources of negative transfer of knowledge.

This model provides a useful general framework for representing the knowledge sources influencing the (task-oriented) user. As such, the properties and content of the knowledge a system user represents, may be regarded as a complex evaluation factor for an interface, which must take into account what the user will and will not know, and how that knowledge will affect interaction with the system. The knowledge sources are shown as exerting influence upon one another in the user’s system representation, although the nature of their interaction is not illuminated.

Characterisation of the Model

In the remainder of the discussion on BIM we need to characterise the model in terms of the principles of usability it addresses, and the evaluation factors accounted for in relation to each principle. In table 2.2 a BIM is represented as capable of ad-
Figure 2.1
A Generic Block Interaction Model

dressing compatibility and UCTDs with respect to the first four evaluation factors, however, its generality and simplicity do not permit it to provide a foundation for addressing performance since it does not cover quantitative or specific qualitative performance metrics. Therefore, without being specific as to the precise form, and content of representations of the user, the application, UI and target tasks, BIMs does allude to such representations as idealised knowledge sources. It assumes that the more "double bounded" knowledge a user has of a UI (either from experience with the system or as a result of good user-centred design), the better a user's performance with the system will be. The model is purely a representation of knowledge sources that might then be addressed by further models of users that would include evaluation or performance metrics.

For the above reasons, this model is not a general model of the user, it focuses only
<table>
<thead>
<tr>
<th>Evaluation Factors</th>
<th>Usability Principles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simplicity</td>
</tr>
<tr>
<td>Users (Us)</td>
<td>Yes</td>
</tr>
<tr>
<td>System Application (App)</td>
<td>Yes</td>
</tr>
<tr>
<td>User Interface (UI)</td>
<td></td>
</tr>
<tr>
<td>Target Tasks (TIs)</td>
<td></td>
</tr>
<tr>
<td>Acceptability of Performance (Acc)</td>
<td></td>
</tr>
</tbody>
</table>
on one part of the user, that is what the user's knowledge sources are, and how the knowledge, from there various sources, represented by the user might be interrelated. The actual form of this knowledge and how it is utilised with respect to task execution are not addressed. It can be argued that a specific BIM may not be a model of a user's knowledge at all, but simply an elucidation of the possible knowledge sources a user might be exposed to together with a weak indication of negative transfer or interference between knowledge sources based upon the differences between the concepts and structures they embody.

Simplicity of the user's representation, or of the UI itself, is therefore not covered by the BIM, since a model that addresses this kind of principle must in some way elucidate the amount and structure of the knowledge coming from the various sources. Consistency, observability and retrievability, which are all principles which require an independent, objective (non-user-) model of the device (as indicated in Chapter 1) are not really addressed by BIMs, since no system modelling techniques, formal or otherwise, are advocated within this framework which might allow us to observe system characteristics that indicate these properties.

The BIM is thus only a reference point from which to begin when addressing what it is that HCI design and evaluative techniques address. The issues that it raises are important to all models which seek to give a complete picture of user knowledge representation. Models which seek to give a comprehensive treatment of the representation of knowledge based upon different kinds of experience must be able to characterise the nature of the differently acquired types of knowledge (represented by the knowledge sources in the BIM) and also show how influence between types expresses itself both in processing and in behaviour (Barnard 1987). The BIMs Framework is by no means a complete framework, and only elaborates a relatively small portion of the Usability Scoping Matrix in table 2.2. Most importantly, it does not address performance aspects which means that it is essentially a research tool which identifies some of the important issues which other more applied models will need to address.
2.2.2 Reisner’s Formal Interactive Grammar

Classification of the Model

Reisner’s Formal Grammar (1981, 1982) was an early attempt to apply a "formal" grammar to the language required to interact with a system (including key presses, mouse movements etc). Referring back to the distinctions between types of HCI models made in Chapter 1; this model belongs to the class of HCI evaluation techniques. It is not a design model because it does not prescribe a method for designing better systems, it only suggests metrics which can be applied to the formal representations of existing design specifications in evaluation. It does not qualify as a performance model because it has no method to associate performance metrics with the measures of complexity which it incorporates so it may only be applied as a comparison technique. It could qualify as a valid competence model because it can describe a set of rules sufficient to generate legal command strings.

The Formal Grammar aims to capture cognitive factors involved in users' representations of interaction grammars. However the notation is only loosely based upon valid psychological principles. It fits more easily into the category of interaction models as opposed to psychological models because the form that the grammar takes is a BNF (Backus-Naur Form) expression. This notation is typically used for system behavioural specifications. It is not based upon what we know about users representations of rules required to use some device.

The Formal Grammar also avoids semantic restrictions wherever possible, in order to maintain simplicity of the interpretation of the grammar. On the other hand this means that the general structures (i.e. resemblances) applicable over similar related types of task which are apparent in a language, and which would be exploited by users, cannot be captured. So that although it succeeds in presenting a production system which might satisfy the competence requirements of the system, any psychological validity claimed for this production system is dubious (Green et al 1985).

Scope of the Model

The focus of the model is what Reisner describes as the action language which is the sequences of button presses, joystick motions, typing actions, etc which the user
performs. Of particular interest in her description are the cognitive factors which are what the user has to learn and remember, rather than physical actions.

Action languages are described in terms of a production rule grammar in BNF notational form; a set of rules of the form \((\text{if the goal is}) \text{ CONDITION} \ (\text{then}) \text{ ACTION}\). BNF can be used to describe generative context free grammars. It unambiguously expresses syntax, but does not readily capture semantics and cannot predict performance quantitatively.

The rules are expressed at a number of levels of detail, beginning with a starting rule, which can be broken down into greater levels of detail until the rules embody the actual terminal symbols (as a notational convention, the terminal symbols are always written in capital letters in the Grammar); these are the 'words' of the interactive language, or the physical actions a user must carry out. Initially Reisner only specified what she called the cognitive terminals. She later enhanced the model to distinguish and model physical terminals (device features utilised), visual terminals (what the user looks at), and action terminals (the physical actions required; which are especially important when modelling repetitive tasks). The general features for the production rule grammar are as follows:

1.) A set of terminal symbols (the words in the language, represented by capital letters);
2.) A set of non terminal symbols (invented constructs representing sets of similar actions that can be grouped together, used to show the structure of the language, e.g. "noun phrase");
3.) A starting symbol (e.g. "S", for sentence);
4.) The metasymbols "+", "|", "::=" (accepted meanings for these are "and" (implying sequence), "or", and "is composed of" respectively);
5.) Rules constructed from the above (e.g., \(S ::= \text{noun phrase} + \text{verb phrase}\)).

Some examples of action language rules described in the Formal Grammar (from Reisner 1981) would be:

\[
\text{picture ::= coloured shape | picture + coloured shape}
\]

\[
\text{shape ::= discrete shape | continuous shape | text shape}
\]
discrete shape ::= separate discrete shape | connected discrete shape

separate discrete shape ::= select separate discrete shape +
                        describe separate discrete shape

The model is structured so as to give an indication of the psychological complexity of an action language. In the first version it uses length of sentence strings and number of rules involved in a BNF model of a language. Two predictions this model makes for an interactive language are:

i) The longer the sentence strings are, the harder it will be to learn and the more errors will be made in using it.

ii) The more rules involved in a language, the more difficult it will be to use.

The action language was tested in an analysis of two graphics systems (Reisner 1981). The model predicted that the more complex device would entail longer learning times and a greater incidence of errors, and it was indeed successful in indicating areas of complexity which were confirmed by empirical observation of users' performance. The model was later augmented to incorporate further metrics (Reisner 1982): the number of terminal symbols of various kinds (e.g. cognitive; action; etc), and total number of rules needed to describe some sub-set of the language. Reisner states that the main advantages of Formal Grammar as an analytic tool are that it encourages precision in evaluation, it generates testable hypotheses about quantifiable, general and intrinsic properties of usable systems, it can be automatically manipulated, and it can provide explanation of user errors.

Reisner admits that better notational schemes than BNF must be found or devised in order to 'reveal both structure and legal strings' although she proposes that BNF can be useful in her view as she says it can fulfill the role of an analytical tool giving early warning of design flaws. It forces the evaluator to be precise about the description of the interface. It allows the generation of testable hypotheses and makes it possible to identify the intrinsic properties of a language which make it easy to learn and use. Also she claims that user errors can be explained on the basis of this formalism. However the level of explanation must, by necessity, be restricted to common sense inferences based upon what the user's known background and understanding of the system are, since cognitive processes are not
modeled or explained. Such inferences may only turn out to be common sense for a psychologist.

**Characterisation of the Model**

Table 2.3 restricts the scope of this model to the principle of *simplicity*. The non-appearance of the model under *compatibility* in the table is not surprising, since it does not appeal to user representations of labels, and actions, which might exist prior to experience with the UI. It cannot therefore tackle issues relating to naturalness of UI representations and behaviour. In fact even going so far as to address semantics of the interpretation of the grammar meant that additional specification of certain restrictions on the acceptable tasks was necessary. For example if the model was to account for the impossibility of describing a circle when box mode had been selected, a semantic restriction was necessary (i.e. an extra production rule was required to stipulate that the shapes selected and described had to be the same).

The Formal Grammar does not explicitly tackle the principles of *observability*, *consistency* and *retrievability*. The model restricts itself to representations of an idealised set of interaction rules, without contemplating the issues of how the device behaviour might influence the user's ability to achieve this ideal state of affairs. When Reisner speaks of "consistency", she means it in a different sense to that defined here. Her definition relates to the number of rules required to carry out tasks with the system, with greater consistency occurring where general rules can be used over a number of tasks. This definition is informal, depending on user or task related factors such as knowledge of expected user goal structures, irrelevant or redundant (but legal) possible rules. This definition of consistency can be interpreted here as *simplicity* which the model is shown as capturing in table 2.3 and which is accepted in this thesis as having both formal (e.g. countable numbers of terminal strings) and informal (e.g. arbitrarily incorporated meta-rules which reduce complexity) components.

Reisner's Formal Grammar does not tackle *UCTDs*. The BNF notation is not used to express users' pre-system experience task procedures, although there is no reason why it cannot be used for this purpose. Were the model extended, within its limitations, to map pre-system experience task representations onto the system task
<table>
<thead>
<tr>
<th>Evaluation Factors</th>
<th>Usability Principles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simplicity</td>
</tr>
<tr>
<td>Users (Us)</td>
<td>Yes</td>
</tr>
<tr>
<td>System Application (App)</td>
<td></td>
</tr>
<tr>
<td>User Interface (UI)</td>
<td>Maybe</td>
</tr>
<tr>
<td>Target Tasks (TTS)</td>
<td>Yes</td>
</tr>
<tr>
<td>Acceptability of Performance (Acc)</td>
<td>Competence Only</td>
</tr>
</tbody>
</table>
specifications, (instead of restricting itself to representations posited for users trained to use either of two graphics tools) it may enhance its power of to predict user difficulties with UIs.

Looking at the evaluation factors in table 2.3, the Reisner's approach is seen as only partially dealing with users in the strict sense, since it restricts its analysis to a competence grammar capable of describing all and only the legal strings required to carry out the target tasks. Issues relating to the system application and output from the UI type are not really dealt with, there is no independent description of the system functionality, states and behaviour. It may be possible to claim that UI behaviour is captured in terms of the input required to cause certain events to occur, since basic actions required are represented by the BNF terminal rules (see the "Maybe" entry in the UI row of table 2.1).

Reisner's approach does not fully address the acceptability of performance cell since it is only possible to predict relative acceptability of two or more UIs using the Formal Grammar (Reisner 1981). However the lack of inclusion of explicit human processing characteristics, prevents the model from being able to predict absolute performance. Table 2.3 shows the acceptability of user performance can only be assessed from a competence point of view; in other words performance predictions are derived from an informal assessment of how difficult it would be for users to acquire and apply the competence rules specified as necessary for the UI in question.

Reisner admits that the BNF notation cannot capture psychologically valid representations with any ease or elegance for the reason that the Formal Grammar is not well suited to dealing with semantics. She states "we could describe all and only the legal strings by a very lengthy grammar which listed each shape, and by then redundantly having a rule for selecting and describing each one. But such a description would lose the general structure - that most shapes are constructed in similar ways. On the other hand we could write a more terse grammar than the one we have presented, showing more of the general structure, but at the expense of more semantic restrictions." There is a trade off between formal precision and semantic interpretation which is a problem with generative grammars which take account of other aspects of linguistic representation and performance than the syntactic rules.
Reisner's Formal Grammar was one of the earlier attempts at modelling interaction, and as such it may be considered crude as a user-representative competence model when compared with later approaches (e.g. Payne & Green 1986). She herself states "in the long run better notational schemes need to be found or devised which reveal both structure and legal strings. Formal Grammar is included here because it raises some of the problems which later techniques have attempted to solve; especially problems with the psychological validity of the grammar, inclusion of cognitive constraints on processing, and representation of device behaviour independently from the user's device representation.

2.2.3 Task Action Grammar

TAG (Payne 1984; Green et al 1985; Payne & Green 1986) is a later and somewhat more sophisticated model than Reisner's Formal Grammar but, in a sense, Payne and Green can be said to standing upon Reisner's shoulders. TAG attempts to extend the powers of 'formal' descriptive approaches beyond those of the Formal Grammar to capture more convincingly the psychological aspects of interaction including general structures which users would capitalise on.

Task Action Grammar (TAG) is a "metalanguage" based upon an earlier linguistic, grammatical representation set grammar. Set grammar was so named because it described rewrite rules which operated on sets of grammatical objects rather than individual nonterminal or terminal symbols. In this respect it chose to trade formal rigour for psychological validity generalizing features of the UI on the basis of similarities between rules. The set grammar and the TAG developed from it generate a representation of a UI action language assumed to be held by the user.

Classification of the Model

TAG is clearly an evaluation technique since it proposes no method for generating the UI it describes. It should be possible to use this model to evaluate existing UIs or detailed specifications, and this is one of the claims that is made for it. It is also claimed that TAG is an improvement on the Reisner's Formal Grammar because it does model users' representations of interactive tasks. In other words TAG aims to represent psychological aspects of interaction. However, there is scant evidence to suggest that users do in fact represent tasks and grammars in the way TAG does.
Furthermore there are no explicit mechanisms in the TAG model to deal with cognitive aspects of processing. The authors state quite clearly "Although we have not specified a learning mechanism, we simply assume one with the appropriate properties", although they do not elaborate what exactly these are. It is probably safer, therefore, to assume that TAG fits more comfortably into the class of interaction models, despite its authors' aim for it to represent a "grammar in the head" of the system user. The fact that TAG does not include any human processing components together with performance constraints means that it is restricted to the class of competence models and as such can only be used, like Reisner's Formal Grammar, for qualitative or comparative evaluations. There is no way of determining absolute levels of performance using this model.

**Scope of the Model**

A TAG description consists of a simple task dictionary in which simple tasks are identified and defined by their semantic components. A feature grammar in which the dimensions of semantic components serve as features (e.g. dimensions of direction would be perhaps up, down, left and right). For more complex examples an explicit list of all the possible features and their values is used possibly to act as a memory aid.

TAG generates action specifications from simple tasks based upon the hypothesis that, in problem solving, complex tasks are organised into a sequence of sub-tasks (Newell and Simon 1972). It uses two levels of description; concepts and rule schemata. A dictionary of concepts is used to model the mental representation of the grammatical objects and simple tasks in the task language. A simple task is defined as one that a user can routinely perform (there is no requirement for a cognitive control structure to modulate its execution). Simple tasks may be very basic or low level) or they may be relatively complex, depending upon the skill of the user. The rule schemata model the mental representation of mappings from task descriptions to action specifications. In addition schemata are required for mapping commands onto their names (in a command language) and for capturing the syntactic rules of the language.

Payne and Green attempt to capture generalizations about the consistency or regularity of a language in a psychologically valid manner (their definition of con-
sistency is similar to Reisner’s). Although they find it difficult to be precise in their definition of consistency as noted earlier in this chapter, Payne & Green (1986) present a list of examples of types of consistency which TAG might capture. These include:

* Syntactic Consistency: termed family resemblance is the consistent use of one expression as a common element in another expression.

* Lexical Consistency: congruence is the matching of lexical (external to the command language) and semantic (internal) relations.

* Semantic-Syntactic Alignment: commonality of organising principles is the consistent mapping of task semantics onto language syntax (this definition overlaps with consistency as defined in this thesis).

* Semantic Consistency: or completeness is the consistency of the extensional semantics of the language, such that types of language and system structure with similar properties (such as real identifiers and Boolean identifiers in Pascal) behave, and can be manipulated, in similar ways.

For a fuller description of these examples see Payne & Green (1986).

The advantage claimed for the TAG technique is that it can be applied to any task-action interface between person and machine, (including lexical command languages, direct manipulation interfaces and knobs-and-dials control panels) and it has mechanisms for relating the syntax of a language to its semantics. Basically what is being claimed is that the structure that this model highlights (by specifying a grammar for the basic tasks and rules on how to carry them out using the system) is the structure that the user might plausibly have in his or her cognitive organisation of the language as he or she learns it.

A Task Action Grammar has three parts; (a) an optional list of features, used only to help categorise the simple tasks; (b) a dictionary of simple tasks; and (c) a set of rule schemas which represent the rewriting of a task into a sequence of actions. A brief example of the form of TAG for a list of commands (from Payne & Green 1986) follows:

**List of Commands**

*Move cursor one character forward .................. ctrl-C*

*Move cursor one character backward .................. meta-C*
More cursor one word forward .................................. ctrl-W
Move cursor one word backward ............................... meta-W

TAG Definition

LIST OF FEATURES (or Feature Sets)

Direction ... forward, backward, right, left
Unit .......... character, word

DICTIONARY OF SIMPLE TASKS

Move cursor one character forward (Direction = forward, Unit = char)
Move cursor one character backward (Direction = backward, Unit = char)
Move cursor one word forward (Direction = forward, Unit = word)
Move cursor one word backward (Direction = backward, Unit = word)

TASK ACTION RULE SCHEMAS ("T " denotes Task)

T [Direction, Unit] -> symbol[Direction] + letter[Unit] (1)

Symbol [Direction = forward] -> "ctrl" (2)
Symbol [Direction = backward] -> "meta" (3)
Symbol [Unit = word] -> "W" (4)
Symbol [Unit = word] -> "C" (5)

AN EXAMPLE OF A SINGLE LEVEL REWRITE RULE (Using Rules 2 & 4)

T [Direction=forward, Unit=word] -> symbol [Direction=forward] + letter [Unit=word]

The list of commands here involve the control of cursor movement. The simple-task dictionary for this small example represents all the routine tasks that the user can perform. The numbers in brackets after each task action rule schema are rule numbers. The rule-schemas generate action specifications from simple-tasks.
Rule (1) is a general task-action rule schema for simple-tasks and is expanded by assigning values to all the features in the square brackets (a feature must be assigned the same value wherever it appears in the rule).

This simple example demonstrates the way in which TAG can describe the task language independently from its actual symbol names. The description models the number and nature of simple tasks available to the user and the rewrite rules which allow the implementation of these tasks using the appropriate symbols.

The example of a rewrite rule, or rule schema, above shows the power of the grammar to generate task sequences from the components of the language without resorting to the actual physical lexemes. This means that one general task-action rule-schema description can capture a set, or family, of tasks with identical or similar structures, capitalising on the psychologically plausible assumption that a generative grammar does not have to have a separate representation for each member of the set of similar tasks as suggested by Reisner (1981, 1982).

TAG formulates a representation of interaction rules to illustrate complexity (or what Payne and Green refer to as consistency) which the user will have to master in the achievement of competence which is what the model focuses on. The main metric of complexity is the number of simple-task rule schemas, discounting the number of rules that are captured by world knowledge schemas mentioned below. Unfortunately there is some debate as to the exact nature of a simple task. When Payne and Green describe such an operation as one that the user can routinely perform, they mean that the definition depends upon the skill of the user. Experts may routinely perform quite complex tasks which would take novices some time to execute, compare the concert pianist’s execution of a scale in a key with many sharps or flats with that of a beginner. However, there is no clear guiding principle as to how to determine what constitutes a simple task for a particular user, and it is certainly not clear from the definition referring to routineness.

The focus on simple tasks is based upon the assumption that these comprise the main determinant of novice user performance. The main advantage of TAG’s metrics over those of the Formal Grammar is the inclusion of family resemblances between the rules as a predictor of increased ease of learning.
The TAG metalanguage is claimed to have other advantages over its predecessors in the form of extra mechanisms for relating the syntax of a language to its semantics. World knowledge and goals represented within rule schemata (in subrules) enable simpler schema to capture wider generalizations on the assumption that certain aspects of the user's representation are independent of the action language and its complexity. Congruence can be expressed by incorporation feature sets related to concepts, such that the model describes how users can generate different names for related concepts on the basis of common features. For example, ADVANCE shares many features with RETREAT, so in the context of having learnt that the command ADVANCE means go forward, the user will assume that the reverse effect will be achieved by specifying REVERSE.

Task structure is dealt with by using subrules, rather like subroutines in programs. The problem with a competence model which specifies task structures for systems where there are more than one way of achieving tasks is that one cannot be sure that the model adequately represents this possible diversity. Competence models are thus distinguished from acceptance models on the basis that acceptance models specify every accepted method, whereas a competence model must constrain itself to the most efficient set of methods for accomplishing tasks in order to model consistency (complexity) even if this representation is not actually held by a user. The hypothesis is that a more consistent action language can in an ideal representation be described with fewer TAG rules than a less consistent one.

TAG is thus a generative grammar modelling the lower bound of complexity of a possible user representation. Presumably, the more unconstrained a UI environment is, the less predictive power such a model can be expected to achieve. In such circumstances acceptance grammars will be required to highlight possible user representations of parts of the action language which can lead to problems. Payne and Green argue that this problem is no less apparent in other models and cite Kieras and Polson's (1985) adoption of task representations defined in a manual, which are only a subset of those users might have.

Characterisation of the Model

From the Usability Scoping Matrix it can be seen that TAG is viewed as a much more extensive technique than Reisner's Formal Grammar. TAG does not incor-
Table 2.4
Usability Scoping Matrix for Task Action Grammar

<table>
<thead>
<tr>
<th>Evaluation Factors</th>
<th>Usability Principles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simplicity</td>
</tr>
<tr>
<td>Users (Us)</td>
<td>Yes</td>
</tr>
<tr>
<td>System Application (App)</td>
<td>Maybe</td>
</tr>
<tr>
<td>User Interface (UI)</td>
<td>Maybe</td>
</tr>
<tr>
<td>Target Tasks (TTs)</td>
<td>Yes</td>
</tr>
<tr>
<td>Acceptability of Performance (Acc)</td>
<td>Competence Only</td>
</tr>
</tbody>
</table>
porate a method for independently specifying system characteristics, and thus cannot be said to capture *Observability, Consistency* (by the definition given for the matrix) and *Retrievability*. However there is an implicit assumption that TAG's could be used to express and compare models of users' system-independent task representations with models of system-dependent representations, although this has yet to be empirically verified.

The "Maybe" entries in the SApp and UI rows of table 2.4 suggest that the TAG grammar is capable of capturing some objective properties of both the application behaviour and of the UI, since it is sensitive to notions of the semantics of system objects and behaviour such as lexical consistency, semantic syntactic alignment and semantic consistency (see above). However this sensitivity is extremely limited (to the effects of simple tasks, rather than whole sequences of tasks) since no independent representation of state behaviour of the system is provided by the approach.

*Simplicity* which Payne and Green refer to as Consistency is thoroughly dealt with by TAG. An explicit model of a user representation of a competence grammar for an explicit set of *Target Tasks* can be generated by TAG. TAG attempts to demonstrate the properties of action languages which make them easier or harder to learn, and to provide predictions and explanation of errors. Since there are no cognitive components in the framework that deal with processing characteristics and constraints, TAG cannot model quantitative aspects of user behaviour.

*System Applications* are implicitly tackled by devices capturing *Semantic-Syntactic Alignment*, where the semantics of the system are related to the syntax of the grammar (this notion overlaps with the definition of *consistency* assumed in this thesis). The UI is dealt with on roughly the same level as it is in Reisner's approach. Characteristics such as presentation, physical actions and so forth are not not discussed by the authors, although it is fair to say that TAG is currently being extended in this direction (Payne & Howes 1989).

*Compatibility* is partially expressed in TAG by generation of fragmentary world knowledge subrules, within simple task schemata, by appealing to common semantic definitions (assumed to be held by most users) for lexemes appearing in the action language. However there is no integrated model of world knowledge which
might generate such representations. Anyone using the TAG methodology would require a certain amount of intuition and psychological expertise to generate the world knowledge rules, and so the matrix suggests only partial treatment of this type of principle.

**UCTDs** (User Centred Task Dynamics) are dealt with by the ability of TAG to express more complex task hierarchies and how they may be represented differently by users (and implicitly by the system) trying to achieve a particular goal. By using a hierarchical structuring device, which involves breaking larger tasks into subtasks which can be organised in different ways (rather like subroutining), Payne and Green (1986) are able to highlight 'organisational consistency and conflict'. This means that TAG has the power to capture UCTDs with respect to users' representations, *system application* behaviour, and with respect to the *target tasks* users may be expected to perform with the system.

Since TAG generates competence models only, it can only claim to achieve partial success in addressing *acceptability of performance*.

Payne and Green (1986) claim that the grammar fits well with performance models because it defines mapping from tasks to actions. It is true that the task rules could have performance factors assigned to their components (e.g. time estimates associated with different types of simple task). However this aspect of TAG has not yet been exploited, there are as yet no components of TAG relating to performance which could be tested and render the model in its totality open to falsification.

### 2.2.4 ACT*

Anderson's ACT* (1982, 1983) is a special case amongst the models discussed here as it has been developed primarily to explain and predict human natural language acquisition and performance. It is however equally valid as a model of the system user's interaction language representation because it embodies mechanisms and cognitive processing constraints which suffice to deal with all of the main processing phenomena associated with language comprehension and generation, to the extent that it has been used to drive computer simulations of language acquisition and use (Anderson 1983). The most obvious distinction between this and the preceding two models is that, whereas Reisner's Formal Grammar and TAG are
both generative grammars, ACT* is clearly a process model, since it proposes mechanisms for representing, storing and processing the information required to interact with a system.

The main reason for including this model here is because it is an important component of one of the few HCI performance models in existence which is the User-Device Model (UDM) and Cognitive Complexity Theory (CCT) (Kieras & Polson 1985) which will be discussed later. The complexity of ACT* itself, and its general validity for other performance models suggest that a fairly detailed description of its nature is best presented separately. However, some of the applications suggested for ACT* are only realised in CCT and are not those proposed by Anderson himself.

Classification of the Model

ACT* is not a design model because it proposes no methods for integrating the characteristics of the user as an information processor into design specifications. However, ACT* is satisfactory as an evaluation model because it can take proposed user task structures, expressed by production rules and, given an appropriate mapping process, highlight areas of user difficulty (such as complex rule structures or decision points) and also predict performance characteristics.

ACT* is categorically a psychological model, however the expressive notation it is based upon (i.e. production rules) can be used to express, amongst other things, the interaction with a computer system. What makes ACT* more than just a notation for writing down the rules applied by human problem solvers and task executors is that it incorporates a model of the behaviour of the different cognitive components which are involved in controlling human information processing; these will be discussed in more detail below.

The nature of the cognitive components of ACT* and of the architectural and behavioural assumptions, which are based upon extensive empirical evidence described by Anderson (1983), gives the model strong predictive power. This power extends to qualitative and quantitative aspects of human language learning and behaviour. ACT* is capable of generating competence rules in the same way that Reisner's Formal Grammar can with its production notation, but it is primarily
designed to be a performance model.

Scope of the Model

Anderson's ACT* is an advanced Human Associative Memory model (HAM) which can be applied to any kind of human information processing. Anderson restricts his discussion to language processing because it is assumed to be more representative of general processing faculties (which may not be the case for specialised faculties such as those of visual processing) and it is readily open to the scrutiny of analysis techniques which are typically used to verify or falsify hypotheses about the nature of the architecture of the human mind.

The processing characteristics of the model are based upon the idea that cognition is the result of spreading activation from excited nodes. The nodes may represent perceptual stimuli or internally generated excitatory influences. For the purposes of prediction and explanation of behaviour it is not necessary to posit the exact physiological structures that underlie this principle (Pylyshyn, 1984). However it is implied that the nodes are generally neurons or groups of neurons and that they are linked by synapses and other neurons which can determine whether the effect one node has on another is excitatory or inhibitory.

Leaving aside the physical architecture, the functional architecture described by ACT* is surprisingly simple. The whole cognitive system is considered as operating on the principles of production rules. In other words these incorporate procedural knowledge to deal automatically and efficiently with a given set of conditions. Production systems are basically condition/action rules. The number of conditions is not limited and increases with the specificity of the rule. However, the number of actions leading from a single production does tend to be limited as productions have automatic control over behaviour and multiple actions leading from one production would tend to result in rigid or unnecessarily elaborate behaviour patterns.
A typical production rule might be of the form:

**IF** it is raining  
and I am going out  
**THEN** take an umbrella

A more specific production rule might be:

**IF** it is raining  
and I am going out  
and the weather forecast says it will stop raining soon  
and I don’t want to carry my umbrella  
**THEN** do not take an umbrella

Of course, these examples are of conscious decisions one might make. Production rules in ACT* are considered as operating at the unconscious level in, for example, word recognition. However the structure is the same at whatever level the rule applies.

The way in which production systems are applied must be controlled since they must be stored somehow and only a certain number can apply at any one time. A general framework is proposed by Anderson which identifies the major structural components of ACT* and their interlinking processes. This framework is illustrated in figure 2.2.

The framework proposed by Anderson provides a testable, theoretical basis for ACT*. The basic architecture of ACT* (which it shares with other production systems) is as follows:

*The Production Memory* contains the procedural knowledge of the productions which can be organised in many ways, for example, according to internal or external conditions or to goals.

*The Declarative Memory* stores declarative items of information, such as "a terrier is a type of dog. Items in declarative memory are semantically related to one another so that, if a subject heard the word 'terrier' in an experiment, he or she would be likely to be primed (able to respond more rapidly) to the word 'dog'. The Working Memory is a database of symbols or conditions to which
Figure 2.2
A General Framework for the ACT Production System,
Identifying the Major Structural Components
and their Interlinking Processes

productions may apply. The Activation Rules associate the appropriate production with the condition symbols in working memory. The Conflict Resolution Rules decide between competing productions which could equally apply. In other words, they can specify priorities of different goals that might be satisfied by each production. There are a number of ways of resolving conflict, some rules which can be applied are:

The Recency Rule: Choose the production rule that applies to the most recently entered contents of working memory. The Specificity Rule: Choose the production with the most specific conditions that are satisfied by the situation. The Refractoriness Rule: Do not choose a production that has just fired even though the
conditions are still appropriate.

This architecture together with a number of processing assumptions is described in some detail by Anderson (1983). The main advantage of this model is that, although it is broad in the issues it addresses, it also has depth whilst remaining relatively simple. The model does not attempt to describe the effects of emotional state or background noise which may have important effects on the processes it does cover. Although in it might be useful to include these as general error factors in a predictive simulation to achieve realistic results, these other psychological factors are not actually part of the processes that control and carry out cognitive processing. To include these factors in a human processing model would considerably increase the complexity of building it. Instead, as Anderson found, it is possible to observe these phenomena and then, knowing a certain amount about their effects, to predict how they would interfere with or enhance the performance of the modelled processes.

One reservation about the processing considerations of ACT* should be included here. It has been a matter of considerable debate (Kosslyn & Pomerantz, 1977; Anderson, 1979; Pylyshyn, 1984; Briscoe, 1987; and others) whether it is necessary to posit a variety of types of representation such as declarative, spatial etc (as Anderson does) in order to model human cognition. Computational models e.g. Briscoe (1987) do not require these additional constructs and there is no neurophysiological evidence for them (Oatley, 1978). In spite of this, Anderson has presented empirical support for the ACT* predictions of language acquisition and so the theoretical disputes can be ignored for the purposes of discussion in this thesis.

Characterisation of the Model

Table 2.5 shows ACT* as being capable of addressing only Simplicity, Compatibility and UCTDs amongst the usability principles. On the other hand it lacks any devices for representing system states so it cannot deal with the other three principles. However it is interesting to note that in this respect ACT* has greater scope than Reisner’s Formal Grammar and the same power as TAG, even though it is not explicitly a model for HCI.
The "Maybe" entries in the UI row of table 2.5 refer to the fact that ACT* productions may be used to model the behaviour of the UI itself as well as that of the user.

The problem with all of these models if they are to be applied to UIs is that they implicitly rely upon some other specification of the behaviour of the device which might indicate what the inputs supplied to the model of the user might be and what the effects of user actions in a given system state might be. If such a specification is not available then the analysis must assume that the UI is well designed in terms of what information it gives the user in terms of prompts, error messages, appropriate options, and how it responds to commands in terms of modelessness, undoability, and so on. If such features are not well thought out, then carrying out user modelling would be rather like carrying out a valuation survey on a heap of rubble.

The model only tackles users, target tasks, and acceptability of performance amongst the five evaluation factors. Since there are no explicit mechanisms for representing system applications, such as Payne and Green's (1986) semantic-syntactic alignment, ACT* will have to be considered inadequate for HCI modelling in this area. There is also no direct discussion in the ACT* literature on the processing and operating advantages, and disadvantages, of the features of different UIs. However ACT* as a generally applicable and detailed model would be very well suited to making behavioural and performance predictions distinguishing between different types of interactive style if only they could be suitably represented for mapping onto the production rule architecture used.

The value of ACT* as an HCI model, when integrated with more purpose built task (Card et al's GOMS) and device representations will be demonstrated by Kieras and Polson's (1985) Cognitive Complexity Theory, discussed later in this chapter.

2.2.5 Goals Operators Methods and Selection-Rules (GOMS)

Card Moran and Newell (1983) produced some seminal work on performance modelling in HCI. The GOMS family of models is based upon earlier work by the authors called the Keystroke Level Model (Card et al, 1980) which is based upon
<table>
<thead>
<tr>
<th>Evaluation Factors</th>
<th>Usability Principles</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simplicity</td>
<td>Compatibility</td>
<td>Observability</td>
<td>Usability</td>
</tr>
<tr>
<td>Users (Us)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>System Application (App)</td>
<td>Yes</td>
<td>Maybe</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>User Interface (UI)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Target Tasks (TTS)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Acceptability of Performance (Acc)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Performance</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Fitts' Law to predict the speed of error-free performance. The keystroke model was aimed at predicting performance times of experts carrying out routine tasks. The work on GOMS was an advance in sophistication on the simpler keystroke model, overcoming some of its limitations. It was used to build simulations of users performing tasks with an interactive system. The GOMS interaction simulations were run in the InterLisp-D environment which is well adapted to simulation purposes.

Card et al were amongst the first researchers to attempt to combine observational data (on how system users carry out tasks with the UI) with an explicit theoretical model; the *Model Human Processor* (MHP), which is a process model of human cognition which, like ACT*, includes performance constraints on processing. The GOMS family of models (family because there can be an infinite variety of individual models which can be expressed at a number of grains of analysis), are methods for describing the operation of an interface in terms of a *state space* which is represented by a goal-stack (goals, subgoals and basic operators or simple actions to achieve those goals). The combination of the MHP and a GOMS model of a UI allows predictive human performance estimates to be derived from a system specification. The GOMS family of models is important because of its ability to make concrete performance predictions, and is relevant, like ACT*, to the work of Kieras & Polson on CCT.

*Classification of the Model*

The MHP and the GOMS family of models of system users task execution require a significant amount of empirical observation of comparable tasks upon which to base performance estimates for task operators of the UI in question (operators are equivalent to Payne & Green's (1986) "simple tasks"). They also need a fairly detailed specification of the system functionality from which to derive the goal structures required to generate an idealised task execution structure. These requirements mean that GOMS cannot easily be used to drive design and belongs in the category of *evaluation techniques*.

Although GOMS does contain a human information processing model, the MHP, which accounts for performance characteristics of perception, memory, cognition and the motor system, the architecture of this model is relatively unsophisticated
when compared to those of Anderson (1983) and Barnard (1986). GOMS does acknowledge the processes of *compilation* and *tuning* which occur as skill increases, and the different levels of rules required for competence, but these are not explicitly accounted for by the model. Consequently GOMS is restricted to idealised skilled-user modelling and is more an *interaction model* than a general psychological model.

The performance predictions derived from applying the processing assumptions to the specified goal structures are time estimates, with associated error factors, which account for variability in real-user performance. The fact that the model generates such predictions means that it can be classed as a *performance model*. But, when a set of tasks has been described, the notation, which in GOMS represents goal-stacks, can be said to express a pragmatic competence model which a successful system user might plausibly acquire as part of his or her representation of how to use the device.

*Scope of the Model*

The two main components of a predictive GOMS analysis are the MHP and the empirically based GOMS family of models of task execution with a device. We will begin our discussion of the scope of the GOMS modelling technique with an overview of the cognitive processing assumptions made. Figure 2.3 is an outline of the basic architecture for human information processing assumed by the HMP which, like ACT* has a number of processing assumptions called "principles of operation" (see Card et al 1983).

The formulae and parameter values for the cognitive processing mechanisms are not included in figure 2.3 as there is not space to give them thorough consideration in this discussion. For more detail on the exact values of the parameters used in the functions for calculating processing times the reader should refer to Card et al (1983). These values are not of prime concern here and it will suffice to say that the predictions generated from the processing functions have been empirically supported by the research described by the authors. What is of interest here is the scope of the model with respect to designing and evaluating UIs in general (the research conducted to verify the predictive power of the GOMS modelling technique has been, for the most part, directed at text editing activities such as
reaching-to-target operations, typing in and replacing text, drawing boxes, getting in and out of files and file handling).

Figure 2.3
The Basic Architecture for the MHP
From Card, Moran and Newell (1983)

Sensory information flows into Working Memory through the Perceptual Processor. Working Memory consists of activated chunks in Long-Term Memory. The basic principle of operation of the Model Human Processor is the Recognise-Act Cycle of the cognitive Processor. The Motor Processor is set in motion through activation of chunks in Working Memory.
The MHP focuses upon the input and output aspects of processing, reflecting its concern with performance times which are heavily influenced by input processing and effector processing. Like the ACT* model, the GOMS architecture distinguishes between long- and short-term memories since empirical research has provided a great deal of evidence on the different roles these play in cognition. However, the MHP does not differentiate between declarative and production memory since such a distinction is outside the scope of the GOMS approach and is not necessary to achieve accurate performance predictions in the laboratory.

The principles of operation described in figure 2.3 are roughly equivalent in their general function to those of ACT*, but they are not as extensive, they operate at a higher level of description, and they do not account for the more detailed characteristics of learning such as reinforcement and weakening of different representations for actions and the compilation and tuning of representations which give rise to more fluid skilled performance. These differences reflect the aims of the GOMS modelling technique which is primarily to predict idealised task performance (of the skilled user) and is not concerned with predicting the areas of performance where errors are likely to occur, nor what the nature of those errors will be.

The GOMS family of models are concerned with the execution of task goals in a hierarchical sequence without accounting for how the alternative competing methods for achieving them might influence performance. GOMS models assume that the correct or most suitable method is always selected. This means that users' mistakes are accounted for only by error factors in the performance predictions. Inefficient users who fail to choose optimal methods are not effectively dealt with at all (Grudin & Maclean 1984).

As mentioned above, the GOMS models can be formulated at different levels of grain of analysis. Card et al (1983) assigned different GOMS models to four levels, although the distinctions between these must be viewed as pragmatic rather than as a reflection of discrete levels of behavioural organisation. The levels are referred to, in increasing order of detail, (or decreasing amount of time it takes to complete operators) as:

(I) The Unit-Task Level
(II) The Functional Level
(III) The Argument Level
(IV) The Keystroke Level

Regardless of levels of detail, all GOMS models share the four key components; Goals, Operators, Methods, and Selection Rules; descriptive features which may be mediated by different cognitive components depending upon the level of analysis (for example, a goal at the keystroke level may be procedurally represented, whereas at the unit-task level it may be conscious and more complex, but the MHP does not distinguish between the two). The following is an outline of the definitions given for these four main components of the model by Card et al (1983):

Goals. (e.g. GOAL: EDIT-MANUSCRIPT): A goal is a symbolic structure that defines a state of affairs to be achieved, and determines a set of possible methods by which it can be accomplished. The dynamic function of a goal is to provide a memory point to which the system (the MHP) can return on failure or error, and from which information can be obtained about what is desired, what methods are available and what has already been tried.

Operators. (e.g. GET-NEXT-PAGE): Operators are elementary perceptual, motor or cognitive acts whose execution is necessary to change any aspect of the user's mental state or to affect the task environment. The user's behaviour is ultimately recordable as a sequence of these operators. A GOMS model does not deal with any fine structure of concurrent actions. Behaviour is assumed to consist of the serial execution of these operators. An operator is defined by a specific effect (output) and a specific duration. The operator may take inputs and its output and duration may be a function of its inputs.

For a specific model, the operators define a grain of analysis (going down to a more detailed grain of analysis might mean that the operator will be represented as a goal). In general they embody a mixture of psychological mechanisms and learned, organised behaviour, the mixture depending upon the level at which the model is cast. The finer the grain of detail, the more operators embody basic psychological mechanisms. Coarser grain analysis gives operators reflecting more the specifics of the task environment, (e.g. keyboard, layout etc).
Methods (e.g. GOAL: EXECUTE-UNIT-TASK
LOCATE-LINE  ...if task not on current line
MODIFY-TEXT
VERIFY-EDIT)

A method describes a procedure for accomplishing a goal, and is a way for users to store knowledge about a task. In a GOMS model a method is a conditional sequence of goals and operators with conditional tests on contents of immediate memory and on the state of the task environment. Methods are always associated with a goal, as in the example, and the operators within the method may have tests associated with them (written in italics as above) so that the sequence above may involve the LOCATE-LINE operator if the current task requires a move to another line. Methods become less likely to succeed, the less knowledge or appreciation of the task environment the user has. This uncertainty is a prime contributor to the problem solving character of the task; its absence is a characteristic of cognitive skill.

Methods are learned procedures which the user already has at performance time; they are not plans that are created during task performance. They constitute one of the major ways in which familiarity (skill) expresses itself. The particular methods a user builds up from prior experience, analysis, and instruction, reflect the detailed structure of the task environment. In manuscript editing tasks they reflect knowledge of the exact sequence of steps required by the editor to accomplish specific tasks.

Control Structure: Selection Rules (e.g. if the number of lines to the next modification is less than three, then use the LF-METHOD; else use the QS-METHOD). Selection rules are all of the form "if X is true of the current task situation, then use method M". Selection rules can be derived individually for given users and for populations of users on the basis of the most common choices in given circumstances. Style rules can be applied to selection rules to produce variations in behaviour which will simulate novice or expert styles (however the error behaviour or failure to complete tasks by novices are not simulated). Selection rules may be written in the goal stacks or listed separately. Their purpose is to ensure that the most appropriate available method to achieve the current goal is smoothly selected without the occurrence of problem solving activity.
The GOMS goal-stack task descriptions are therefore composed of Goals with associated Methods which are made up of a series of Operators. Selection Rules can be used to determine which method should apply if there is more than one, and a probability of a given method being applied can be incorporated in the simulation to match the variability of real users' choices of equally valid (if not equally efficient) methods. The notation itself is rather LISP like, and this is a reflection of the InterLisp software environment within which the GOMS models were experimentally derived and developed.

From their experiments the authors found that, on the whole, the GOMS technique predicted users' choice of methods correctly about 80% ~ 90% of the time; and predicted the actual operators in sequence around 80% ~ 100% of the time. However the Keystroke Level model was only 50% accurate for operator occurrences. The GOMS/MHP model made what the authors considered to be reasonably good time predictions for text modifications; its estimates were within 35% of actual times achieved by users. In general the evidence suggested that accuracy of the GOMS technique reduces as the grain of analysis becomes finer.

As stated above, the GOMS technique is not designed to capture the processes and conditions which give rise to errors, but the GOMS model can be extended to account for errors causing variations in performance times (which is the main GOMS prediction which errors effect). The analyst has to calculate the sequences of goals which will be elicited upon commission of an error and estimate the time taken to execute them. With routine errors it should be possible to add a time factor to accommodate their occurrence, however for non-routine errors such as might occur with less than expert system users, GOMS formulations tend to become highly inaccurate.

Weaknesses of GOMS include the poor treatment of user errors which are not qualitatively anticipated by the analysis. There is a great deal of confusion as to how the appropriate level of model can be determined prior to an evaluation. There is also uncertainty as to what a unit-task really is (i.e. how does one arrive at a sensible set of equally complex units of behaviour when they might all be differently composed). Selection rules are a poor attempt at modelling the user's organisation of behaviour, they do not display the elegance of Anderson's competing productions mechanism, nor do they have the explanatory clarity of the task rules
of Payne & Green's competence grammar.

Characterisation of the Model

The Usability Scoping Matrix for GOMS suggests that, for all its detail, the scope of the GOMS technique is rather limited when compared to others. The only usability principle it captures is simplicity. Since GOMS does not account for the knowledge which users may have compatibility is not addressed. However, it is possible that selection rules for the actual task methods may be empirically derived from observations of real users, in which case GOMS may be capable of addressing UCTDs (hence the "Maybe" entry in the UCTDs column). The only evaluation factors GOMS accounts for are UI, target tasks and acceptability of performance. The system application itself is not simulated, nor are the possible states imperfect users could get into, since only skilled performance is modelled. It is worth pointing out that the GOMS specifications of users tasks can be seen as strictly formal in the logical/algebraic sense because it can be run as a computer simulation.

The GOMS technique only accounts for the cumulative time taken to execute tasks with system UIs and the operators which will be selected in order to do this. In this sense it can be said to address simplicity; a more simple system should permit a given set of tasks to be executed more rapidly, using fewer operations than a more complex one. The other usability principles are not tackled so adeptly by the technique. Observability, Consistency and Retrievability principles would only be addressable via some objective representation of the UI, independent of the user's view (rule based or task oriented). Such a representation is not part of the GOMS technique.

The limited scope of GOMS is partly a result of a trade off problem between generality and specificity which is common to many formalisms. The more precise a model becomes, the greater its size and complexity, hence the less time there usually is to cover all of the ground that might be desirable. GOMS is indeed very precise (regardless of how correct it might be) about the processing characteristics which underlie skilled performance. However the performance which is being modelled is not realistic in the sense that it is idealised and error free, and there is no representation of the actual structures in the users head which might give rise to
Table 2.6
Usability Scoping Matrix for GOMS

<table>
<thead>
<tr>
<th>Evaluation Factors</th>
<th>Simplicity</th>
<th>Compatibility</th>
<th>UCTDs</th>
<th>Observability</th>
<th>Consistency</th>
<th>Retrievability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users (Us)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Application (App)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Interface (UI)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Target Tasks (TTs)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acceptability of Performance (Acc)</td>
<td>Yes Performance</td>
<td></td>
<td></td>
<td>Maybe</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
goal accomplishment. Only the goal hierarchies requiring achievement, together with the necessary operators, are specified. This approach gives the analyst no power to determine human learning and error characteristics likely for a given UI design.

The nature of the system application which underlies necessary features of the UI, such as response delays, parallel activities, etc, is not accounted for by GOMS. The empirical evaluations of the models were carried out on editing systems which have very little application based necessary behaviour, and the GOMS models regard all human actions as sequential, so they could not necessarily account for certain types of application such as monitoring where users may be able to carry out more than one task at a time. Since GOMS does not include a model of system states and show how what the user does affects these states, it cannot be said to address application issues.

The MHP does account for different time characteristics of various types of operator. A keystroke level GOMS model can differentiate between mouse clicks, typed commands, pointing actions, looking, and so on. This means that GOMS has the power to differentiate between systems with different types of UI which may have different input and output styles.

Target tasks are what GOMS focuses upon; in fact it assumes that these are ideally performed by a user, so in a sense GOMS addresses the best possible performance for a given UI. If users can be trained and motivated to some criterion time and error-rate standard, then a GOMS analysis can be a very powerful and accurate tool if it can be applied to a specification which is accurate enough and for which comparable operators can be found from which to derive the basic performance parameters. once these requirements have been satisfied, it should be possible for an analyst to derive relatively reliable time predictions for specified tasks and to determine whether these are acceptable for the system host's purposes. Unfortunately this set of circumstances is quite rare outside the military forces.

2.2.6 Cognitive Complexity Theory

Cognitive Complexity Theory (CCT) (Kieras & Polson, 1985; and Polson & Kieras, 1984, 1985) is a multi method UI analysis technique. The authors describe
it as "an approach to the formal analysis of user complexity".

Their approach includes specifications of the user's *job-task representation*, the *user's goal structure*, and the *device representation* (an extended form of state transition diagrams, e.g. Arbib, 1969; Parnas, 1969) which is an explicit specification of the system behaviour which the user manipulates. The other main features of this approach are a human information processing model which is Anderson's *ACT production systems model* as described in Anderson (1982) which is used to constrain the user's how-to-do-it knowledge; and procedures for generating models of the user's *goal structure* knowledge and *mapping* this onto a *device hierarchy* representation (derived from the transition networks) to highlight *complexity* of the representation a user would require to operate the device.

*Classification of the Model*

The authors of CCT claim that it can "provide a powerful *design methodology* for new devices" (Kieras & Polson, 1985) by allowing designers to "develop simulated prototypes of the device, and a specification of the knowledge required to operate it, before going on to the development of actual hardware and software prototypes". This claim is too strong, since CCT does not actually include any methods for reducing system complexity (e.g. procedures for; reducing the number of different user-operations; reducing command string length; improving reusability with detuning and diffusion of UI functional components).

CCT has components that model both *psychological* aspects of system user behaviour and *interaction* aspects, it also includes the additional system behavioural component. It is probably safer to say that CCT is more targeted at interaction than at users processing behaviour, however its inclusion of ACT* production systems processing and cognitive architecture make it a much broader approach than others described in this chapter. In addition CCT is clearly a *performance* model which includes cognitive processing in its scope. It is better able to deal with representation and learning than GOMS because it uses ACT* instead of the cruder MHP of Card et al (1983).
Scope of the Model

The four main components of the CCT approach are, in logical order of application, as follows:

1. A human information processing architecture based upon the ACT* production systems model with appropriate theoretical performance assumptions.
2. A GOMS representation of the production system model of the user's how-to-do-it knowledge of the task- and device-specific aspects it embodies. This specification includes goal structures, operators, methods, and selection rules whose performance is constrained by (1); this is defined as the user's job-task knowledge representation.
3. A Generalized State Transition network representation of the device.
4. A mapping between the device hierarchy derived from (3) and the user's goal hierarchy derived from (2).

Each of these components will be discussed in order to demonstrate how the CCT approach works.

(1) Production systems architecture of human information processing

The advantage of using the ACT architecture is its adaptability to all linguistic and communication tasks, which means that the applicability for CCT will not be constrained by inadequacies in its cognitive processing model. This may not be true of GOMS (for example the MHP will not deal with naive users because it cannot predict or explain most learning phenomena).

As with all production systems, CCT uses IF (condition) THEN (action) rules to describe the appropriate behaviour for all conditions included in the task specification. This is assumed to model the knowledge that the user has of how to do the tasks with a device.

(2) Notation for describing user's job-task representation

The GOMS notation, used to represent the user's production rules for job-task knowledge, has many LISP-like features and expresses production rules easily. Each production rule includes five terms which are, in order; a name for the rule; the word IF, the condition(s); the word THEN; and the action(s). An AND func-
tion is used to specify that all elementary conditions must be satisfied for the production to fire. Conditions can be tests for patterns in memory or for states of the task environment which can be specified in the individual simulation. Variables in conditions and actions can also be specified in individual simulations.

This notation is used to express both device-dependent and device-independent knowledge because the user would need both to complete the tasks using the system.

Figure 2.4
Examples of Selection Rules in a Job Representation for Editing a Manuscript

(Journal-article
IF (AND (TEST-MSS manuscript is a new journal article)
       (TEST-GOAL type manuscript)
       (TEST-GOAL select equipment))
THEN ((ADD-NOTE many revisions will be done)) )

(Use-Displaywriter
IF (AND (TEST-GOAL type manuscript)
       (TEST-GOAL select equipment)
       (TEST-NOTE Many revisions will be done)
       (TEST-MSS manuscript is long))
THEN ((ADD-GOAL use Displaywriter)
       (ADD-GOAL type new manuscript into Displaywriter)
       (DELETE-GOAL select equipment)) )

An example of the notation is described in figure 2.4 (from Kieras & Polson, 1985) which describes the production rules required by a secretary with the task of typing a clean first draft of a manuscript of a journal article in order to select the "Displaywriter" editor. The example shows how conditions can be tests for goals
as well as other states, and how actions can be to add or delete goals which will then be acted upon by other production rules. The notation also includes NOT conditions (where the production only fires if the specified state is not the case), and special control productions which ensure that productions do not repeat indefinitely. For more detail on the nature of the GOMS representation of the production system in CCT the reader should refer to Kieras & Polson (1985).

The resulting user's job-task representation, together with an appropriate device specification, can be run as a computer simulation controlled by the architectural and processing constraints of the ACT* style human cognitive model. Polson (1987) reports research on a number of simulations of CCT in which the performance, learning and transfer assumptions of the model are experimentally tested. Although certain processing performance functional assumptions may not be as good as they could be, the model is, according to its authors, satisfactorily accurate in all of its areas of prediction.

Figure 2.5
Goal Structure for Secretary
Selecting Device for Editing a Manuscript

EDIT MANUSCRIPT

NEW JOURNAL-ARTICLE

TYPE MANUSCRIPT

SELECT EQUIPMENT
Essentially CCT has been shown to yield reliable metrics which "provide a definite and quantitative characterisation of the user complexity of the device", along with predicting quantitative aspects of performance. The actual complexity of the representation is clarified by extracting goal structures for users tasks with the system. The goal structure for the Journal-article rule example above might be as illustrated in figure 2.5:

The goal structure for deleting a text string string X using the IBM Displaywriter might be as shown in figure 2.6:

Figure 2.6
Goal Structure for Deleting a String of Text

Kieras and Polson derive such goal structures by examining the GOMS production rules and noting where goals are asserted, deleted, or tested. The structures are processed in a depth-first, left-to-right order so that in the delete-string structure the order of goal fulfilment is; POSITION CURSOR; MOVE CURSOR; SELECT RANGE; DELETE-STRING.

It is possible to evaluate the completed goal structures for their complexity; in-
increased numbers of goals indicate more production rules which will involve more processing steps; and increased depth of goal hierarchies means that working memory, which must hold the currently unsatisfied goals, will be more loaded. The authors also note that, using the assumptions of the ACT model, the automation of processing as skill increases will be characterised by increased speed and ability to deal with more complex goal structures. This is predicted by the model due to the lessening of working memory involvement as a result of automation, which means that working memory constraints have a variable effect on performance, becoming less severe as skill increases.

The notation described also generates testable predictions about the character of learning with respect to alternative goal structures and production rules for a given set of tasks. For example, according to the processing model, a general procedure learnt by system users for deleting strings on a particular system would be more difficult to learn, if it required several production rules, than an equivalence of specific procedures which only required one or two rules each (totalling less than the general procedure). It would also not lead to high levels of performance as quickly as the specific rules.

The production rules used in the user task representation model may be written in many alternative forms in order to produce similar output. Kieras and Polson use style rules (as do Card et al. 1983) in order to model variations in task approach between types of individual, particularly for novices who are assumed to go through steps such as testing goals in working memory, explicitly attending to feedback, verifying execution each step, and executing only a single control action in each production.

So the production based user knowledge representation is intended to model the complexity of the knowledge required to execute some set of tasks using some sort of device. Kieras and Polson claim that CCT is able to predict learning times, the generality of knowledge (i.e. how similar it is to device independent knowledge that a user would have) and actual performance times.

Their work has been criticised however by Foss and De Ridder (1987) on the basis that they do not adequately capture the structure of knowledge in their assessments of learning and transfer. The similarities between hierarchies of production rules
which make up goal structures, and which themselves are transferable knowledge, are not captured by CCT. It merely counts number of new productions required by a device user rather than the goal and sub-goal structures within which they are embedded, and how well these reflect existing structures within the user’s head. A new production associated with a high level goal may be more difficult to learn, or have wider impact upon performance than a low level production.

(3) Representation of device behaviour

The behaviour of a UI can be represented implicitly through the GOMS notation specification of how to accomplish tasks using the device; the authors call this how-to-use-it knowledge. Unfortunately having only this type of knowledge prevents a user from generalizing what is learnt to new situations, and will not permit the user to reconstruct anything which has been learnt and then forgotten, but system users obviously do generalize and reconstruct what they have learnt, which implies that they have a how-it-works representation of the device. Norman (1982) distinguishes such users’ representations from ideal representations of how a device works. The user’s conceptual model of a device is purely a user’s meaningful explanation of the behaviour of the device which may, or may not, be an accurate representation.

Kieras and Polson use an idealised formal how-it-works representation of the device which might be presented to users to improve their understanding of how to use it. They stipulate that this representation must only include information which is relevant to the user’s task(s). How such information can reliably be identified is not made clear.

The representation can be viewed as a hierarchy of descriptions with each level in the hierarchy elaborating on the level above. The aspects of the UI that actually require elaboration may be determined by relating the to-be-explained device behaviour to the user’s goal hierarchies. If the user already has a goal as part of his or her task representation then no explanation will be necessary (i.e. for device-independent or previous-device-dependent task knowledge). If the goal is new or only specific to the new device, then that goal must be described by a corresponding level in the device hierarchy. If a possible description does not correspond to a user’s goal then it must be omitted.
Device behaviour is captured by Generalized State Transition Networks (GTNs) which are graphic representations for finite state machines (finite state machines are a general formal way of describing systems) and are the selected device representation notation for CCT. It is from these that suitable device hierarchies for device behaviour explanation are generated. The device hierarchies are also used for mappings from the user’s goal hierarchies to predict and explain user difficulties with the device.

GTNs have several advantages over simple transition networks and augmented transition networks (ATNs) for the purpose of modelling how-it-works knowledge. Notably, although GTNs are equivalent in power to ATNs (Woods 1970), they include extensions which allow them to simplify representations such as parameterizable recursion, and use of nested subnetworks (Cockton 1988) and always ensure that the system is able to make a successful transition, whatever the conditions that prevail. Nesting, which can occur within actions, conditions and states of the device, is a particularly useful concept for the how-it-works representation required by CCT as it permits states and actions that do not concern the user (i.e. being more concerned with application behaviour) to be ascribed to some nested subroutine which does not rely on user input. Nesting is also useful for representing modes of the system which frequently impinge on its behaviour as perceived by the user.

The components of a GTN are labelled nodes connected by directed arcs which may be labelled with conditions (usually above) and actions (usually below the arc). The nodes represent states and the arcs represent transitions between states (transitions require the conditions, if there are any, to be satisfied and if they are, or if there are no conditions, then any actions (there may be none) under the arc will fire.

Figure 2.7 shows a transition network for deleting a string using the same method as represented by the goal hierarchy above (POSITION CURSOR; MOVE CURSOR; SELECT RANGE; DELETE-STRING)

The nodes may be expanded if necessary, for example the SELECT START node may be expanded to show movement and a mouse click if further explanation is required. As can be seen in the network, the system safely exits from delete mode,
even if the target range selected is empty. It should be noted that the GTNs used by Kieras and Polson are, in fact selective models of UI behaviour, rather than underlying application state behaviour, and as such have been criticised as inadequate simulations of true device behaviour (Green et al 1988); this point will be raised again later.

The GTNs derived from a formal analysis of the UI can be simplified, as were the GOMS production representations, into a device hierarchy, or structure graph. This graph extracts the "hierarchy of network embeddings" or states and substates that the system is in while it waits for user input. This hierarchy is the counterpart of the user's goal hierarchy with the system represented as an available sequence of options within some task oriented mode.

(4) Mapping between the user's goal hierarchy and the device hierarchy

The final part of the CCT analysis involves mapping between the device hierarchy and the user's goal hierarchy for each task. This procedure involves drawing arcs connecting isomorphic nodes in the two hierarchies.

Figure 2.8 shows a mapping between a device hierarchy and a user's goal hierarchy for a delete string function (from Kieras & Polson 1985). The authors claim...
Figure 2.8
Example of a Task-to-Device Mapping Between the Device Hierarchy and the User's Goal Hierarchy for Delete String
that the mapping is able to clarify mismatches between the two hierarchies which would be a source of difficulty for the user. The task-to-device mapping shown in figure 2.8 is not good because the user's DELETE STRING goal does not map onto the part of the device structure that one enters upon pressing the delete key. As the authors point out, from the hierarchies it is demonstrated that, if the user has the goal of deleting a string, the first subgoal is to move the cursor to the beginning of the string. Then the user can press the DELETE key to enter the delete mode of the device.

This means that if we accept the goal- and device hierarchies and the mapping procedure we can accept the authors claim that pressing the DELETE key does not correspond to the assertion of a deletion goal. Instead this key-press is deferred until until the user has positioned the cursor. There would be extra strain on working memory, which would have to process the cursor moving activity before the delete goal could be satisfied. This would be sufficient to allow the model to predict more user difficulties with this structure than with a system which avoided this problem.

General Utility of CCT

The CCT analysis produces cognitive complexity assessments based upon the specifications built up of the user's knowledge and the device behaviour. Kieras & Polson (1982) state that the complexity of a device is determined by the complexity of the knowledge required to operate the device. They state that the complexity of a device is given by:

(1) The complexity of the user's task representation, and the learning, memory and processing capacity demands implied by the task representation.

(2) The number of device-dependent functions, which are not part of a user's initial task representation, and the difficulty of acquiring them.

(3) The ease with which a user can acquire how-it-works knowledge.

These aspects of knowledge and behaviour are what CCT aims to model and predict for the designer. Novice and expert users can be modelled with appropriate empirically based assumptions about their processing characteristics which will allow the areas of complexity of a prototype UI to be identified. The prototype can then be altered to reduce the complexity of its operating rules.
The device-to-task mappings generated by the approach mean that CCT may fulfil three different roles for the UI designer. Kieras and Polson claim that it can provide quantitative performance predictions for training time, transfer, and productivity which can be used as early feedback into the design prototyping. They also claim that it can identify and explain problems arising because of poor mappings between users’ goal structures and the device behaviour, and finally, if the device cannot be redesigned, the CCT device hierarchy representations can be used as a basis for training material to enhance users’ how-it-works knowledge of the device which should improve their ability to learn and use the device.

**Characterisation of the Model**

Since the CCT approach is a synthesis of three modelling approaches, it is not surprising that it addresses more issues than any of the other techniques discussed in this chapter. The breadth of the approach means that, particularly for the ACT* based processing architecture and the GTN representations, there are still considerable refinements to be made both in terms of theoretical assumptions (such as the characteristics of working memory) and in terms of methods used to apply CCT (such as the method for nesting GTN device representations). However, these types of problems are less criticisms of the approach than indications of requirements for further work.

The table shows that, although CCT may claim to incorporate a device model, it does not in fact qualify as dealing with application issues. This is because the device model is based on GTNs which are selective representations of the UI or virtual machine behaviour (Green et al 1988). Kieras and Polson make no attempt to ensure that their device model is an accurate and formally correct representation of the whole application’s behaviour; this would involve a great deal of additional effort and they are not interested in the formal properties of the whole system. However this means that the model is never capable of accurately predicting the system’s behaviour, and perhaps deals only approximately with UI behaviour.

CCT deals with all the evaluation factors relating to *simplicity* except the system application, although this does not imply that it does so perfectly. It models the simplicity of user representations, system UI behaviour, target tasks, and makes performance predictions which could be compared with specifications of accepta-
<table>
<thead>
<tr>
<th>Evaluation Factors</th>
<th>Simplicity</th>
<th>Compatibility</th>
<th>UCTDs</th>
<th>Observability</th>
<th>Consistency</th>
<th>Retrievability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users (Us)</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>System Application (App)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Interface (UI)</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Target Tasks (TTs)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Acceptability of Performance (Acc)</td>
<td>Yes Performance</td>
<td>Yes Performance</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
bility. The simplicity of the UI is implicitly dealt with, as in GOMS, because the physical actions required to operate the device are accounted for in the simulation time predictions (so, for example, the model could predict that pointing at a menu item would be simpler than typing in a command string).

Compatibility with respect to users could be addressed by the production system which can (in theory) simulate characteristics resulting from lack of experience with concepts and procedures. However this aspect of knowledge is not explicitly dealt with by CCT which does not account for the effects of users’ having or lacking certain concepts or their knowledge and understanding of interaction command names. CCT concentrates more on the structure of tasks in terms of goal hierarchies and deals much more closely with UCTDs. UCTDs are captured by the user’s representation of target tasks (which is the user’s goal-hierarchy) and the mappings between this and the device-structure.

CCT does rather better than the other HCI techniques on the more formalizable usability principles. It can show whether the information required to perform a target task is observable by including it in the GTN state representation in the form of a component of a state node and a user’s production condition. Unfortunately it does not address whether the information really is the kind that the user needs since there is no way of telling this from a user simulation which always assumes that processing is sensitive to the conditions arbitrarily included in the productions. In other words, since the look and feel of the system are not captured by the model, it is difficult to simulate the true conditions which the user relies upon in processing. Furthermore the GTN nesting may hide important information about system behaviour which could ultimately influence user behaviour. True observability of the application would be very difficult to determine without having more rigorous procedures for deciding what aspects of the device do and do not concern the user. It would be unlikely that valid performance predictions based upon assumptions about the visibility of data could be generated without augmenting the perceptual processing components of the human information model to deal with the relationship between information presentation on screen and human perceptual processes.

Consistency of the UI with respect to the target tasks can be captured easily by the user production system by the rules for acting with the device. If a condition which represents a command name appears with other conditions and different ac-
tions in other production rules, then the commands are clearly inconsistent; the consequence would be predicted increases in user learning and performance times. State transition networks in CCT are not used to model application consistency because they only relate to the UI.

The use of the GTNs allows CCT to model the UI states and the actions possible within them. This means that it should be able to address to some extent whether the virtual system really behaves in the manner that the command names suggest it will; however, such issues would be difficult to tackle without having more extensive GTNs modelling more than a simplified task-oriented UI behaviour of the device. Therefore the treatment of consistency is perhaps partial, but it is certainly possible with CCT.

Finally, with respect to retrievability, CCT does have the potential to show that the interaction simulation can in the course of the target tasks can get the device into undesirable states from which it is difficult to exit. This can be shown using the mappings between the device structure and the user's goal hierarchy. This is not a reliable feature of CCT since the incomplete device specifications advocated would be likely to miss potential irretrievability which may relate more to the nature of the application's necessary functionality than to UI software. Since CCT only addresses idealised task executions, irretrievability could not be simulated as part of a non-perfect or naive users' performance with the device.

2.2.7 Interacting Cognitive Subsystems

Barnard (1986) states that there is a trade off between the depth and breadth with which user models are generally expanded. Consequently models tend to concentrate on certain aspects of the human processor, leaning either towards the cognitive, general problem-solving aspects of behaviour or towards the more rigid faculties of memory, perception, attention and so on. Further, many models are restricted to the controlled conditions of the laboratory. "Move outside the scope of the defined paradigm and predictive power is lost - either because the theory is inaccurate or because it is unclear how a prediction should be arrived at in the novel context." This is a common complaint about theories that are based upon the objective rule structure of the system. Barnard criticises this selectiveness and his model, Interacting Cognitive Subsystems (ICS), is an attempt at unifying the dif-
ferent strands of empirical evidence with a theory that takes a holistic view of the brain as an information processor.

This model is not intended to take over from all the other different types of approaches. Rather it is designed to guide the selection of an appropriate type of model to apply in any particular task situation. As Barnard states, it is commonly thought that different tasks place different demands on the system user. Hence different mental processes will be important in these situations. The problem is deciding what kind of model is most appropriate, or which model concentrates on those aspects of the user which are most involved in the task in question. In addition the model chosen should be capable of taking into account the "knowledge-based processes which construct and manipulate the content of mental representations of meaningful material". The goals set by Barnard are ambitious. Thus, it is implied that a great deal of future work remains to satisfy them.

Research so far has uncovered a great deal of information regarding the quantitative and qualitative differences in user performance in different situations. To try to capture this richness of observed behaviour, Barnard has tried to adopt a broader approach, sacrificing specification in depth and concentrating on a framework to guide selection or design of more specific models which might be applicable. ICS specifies a "distributed processing architecture for human perception, cognition, and action". The nature and properties of the mental representations are specified together with the actual processes that transform one kind of mental representation into another.

Classification of the Model

Like the BIM (Morton et al 1979) Barnard's (1986) ICS reflects growing concern about the potential problem of not knowing what will be the most appropriate analytical HCI approach for a given design situation. ICS is intended to provide "some principled basis for knowing which model is likely to be the most appropriate for which set of circumstances". Therefore ICS cannot be classed as a design or an evaluative technique, although it is intended to support design and evaluation of UIs. ICS, like ACT* is presented as a model that could have implications in fields other than HCI, such as foreign language learning.
ICS is primarily a sophisticated psychological model of a production system architecture and as such aims to simulate the same processes as do the ACT* architecture and the MHP architecture represented in figures 2.2 and 2.3. However Barnard adds further organisation to the model by assuming that all human information processing, for all faculties, comes about as the result of the interplay between a number of processing "functionally independent subsystems" which cover perception, cognition and action, and which communicate over a data network. This psychological model was used to predict some of the problems users had with a specified computer dialogue language (Barnard 1985).

The model is clearly a process model which is capable of generating qualitative performance predictions together with detailed predictions about the characteristics of learning and behaviour of system users. For this reason it is different from the other performance models discussed in this chapter because the focus is not on predicting idealised performance characteristics only. At present there exist no models which give a satisfactory account of all aspects of performance. However ICS would be a contender for such status if it were augmented to include explicit processing parameters for the mechanisms involved in the architecture.

**Scope of the Model**

The framework proposed by Barnard consists of a collection of functionally independent subsystems. Each of these has constituent secondary processes that operate within what Barnard describes as larger configurations of cognitive resources. Each subsystem deals with a different type of representation of information, e.g. acoustic, propositional or limbic, enabling the overall system to encode input, make sense of it, and generate a response to it. The processes must operate in a co-ordinated manner within these configurations. Co-ordination is achieved by communication via the data network which allows the transmission of codes of output from the subsystems. Each subsystem is able to accept certain codes only and can recode information received so that it will be communicable to other subsystems. In this way sensory information can be translated via a series of subsystems into semantic information, and semantic processing can lead to motor (physical) output.

Figure 2.9 gives an overview of the architecture proposed in the ICS framework.
Figure 2.9
An Architecture for Perception, Cognition and Action
(from Barnard 1985)

Figure 2.10 is a more detailed picture of the structure of an individual subsystem.
A brief description of this subsystem architecture is as follows: Each subsystem incorporates secondary processes that transform the input information in a mental representation into an output representation. These processes embody procedural knowledge, as do the hypothesised processes in the ACT* and MHP architectures described earlier. Procedural knowledge enables a system to carry out useful transformations in information processing, and Barnard points out that it can generally only be expressed by its application, being automatic and fixed in nature. In addition each subsystem includes an image record. An episodic trace of representations input to that subsystem is created in this record via a primary copy process. The actual data undergoing recoding by the secondary processes may be from direct input to the subsystem or from information held in its image record. Barnard does not include a general purpose working memory, shared by the different resources, and this is one of the major differences between this and other architectures (Anderson is committed to general purpose structures and processes; Anderson 1983 pp 3 - 5). Barnard also rejects any requirement for a central executive controlling the pattern of activity. The subsystem framework is self controlling by virtue of the data network which allows communication and coordination to take place between the different subsystems. However he does contemplate the possibility of the existence of some general purpose faculties for higher level processing such as planning.

The features common to all subsystems are shown in figure 2.10. The image record is an episodic memory structure which has its main input from the primary copy. The primary copy creates a record of all input to the subsystem from the data network which is then represented in the image record. The secondary processes can then act upon either the data stored in the image record or on direct input from the data network. This arrangement allows the subsystem to be flexible in its processing because the image record acts as a buffer store of data on which it can operate.

The record code I represents the particular code to which the subsystem is sensitive. The information encoded in it can be recoded by the secondary processes into codes which will be picked up by other subsystems. The secondary processes can be thought of as "functionally independent production systems" (a production system is a set of rules of the IF, THEN variety).
Information processing principles are applied to constrain the operation of a subsystem. Secondary processes can, for example, only process one data stream at a time. This inflexibility can be compensated for somewhat by the buffer arrangement of the image record, but the model would assert, therefore, that in a dichotic listening task, a listener would only be able to represent the lexical content of one of the speeches.

The outcome of having the primary copy produce a record (the image record) of all input to a particular subsystem is that the secondary processes may operate on either direct input or in information in the image record. This parallelism gives the ICS three significant processing properties:

1. Two modes of processing (i) secondary processes directly recode input to the subsystem (ii) buffered processing, in which a secondary process operates in series with the primary copy process.

Buffered processing is more robust and is associated with conscious attention to processing activity in the relevant domain.

2. Memory retrieval from the image record permits recoding, by the same process resources, from past input as well as from one of the forms of immediate processing.

3. Learning can be explained by the model. Since the primary copy process is
organised in parallel with (instead of after) the secondary copy processes, it is able to represent an image record for which there are as yet no appropriate recoding procedures. The main function of this episodic memory record is to provide the basic data from which the appropriate procedures develop.

Once the secondary processes have recoded the information it is passed back out into the data network. Any data that passes into the overall system is thus processed in different ways to varying degrees of abstraction by a set of modular systems which are functionally independent of each other but which share a communication network and have many features in common.

The ICS framework has been shown by Barnard to predict various difficulties which users experience in HCI. ICS does not generate quantitative predictions because it does not contain any processing parameters at present. Instead it predicts qualitative characteristics in behaviour and differences between behaviour given different processing demands in learning and using interaction languages. The general implication is that the nature of the errors made by users reflects that of the subsystems from which they originate. Various experiments were designed by Barnard to highlight failures in particular subsystems and the predictions made by the model were validated. There is not space here to go into detail on the additional assumptions which give the model some of its strength. These are described at some length in Barnard (1987).

The ICS framework does assume the existence of certain general resources such as long term memory which can be drawn upon by the subsystems in this framework, particularly in the processes that translate back and forth between the propositional and the implicational subsystems. These general resources are assumed to be capable of planning and organising high level information. This arrangement perhaps leaves something to be desired. The model does not adequately describe or explain the precise nature of the organising mechanisms which involve propositional complexes which can be manipulated in the light of prior experience.

The implicational subsystem is the most abstract and refined data processor and is required to process information in a much more flexible manner than the other subsystems as it must govern the flexible goal directed behaviour of the whole system. Long term memory is used to draw similarities between past and present situa-
tions, problem solving heuristics are applied, goals are held in mind and behaviour is planned and structured. This arrangement is necessary for the simplicity of the model. It is difficult to conceive of a modular implicational system being capable of carrying out all of these functions whilst maintaining the simple structure of all subsystems as shown in figure 2.9.

ICS refers to the interplay between the propositional and implicational subsystems. This is proposed as being the means by which the overall system can organise the processed data available to it. However no formal structure for this mechanism is offered as it could imply the existence of a control structure of some sort, which is precisely what ICS seeks to avoid. In fact ICS is not committed to any particular theory of speech perception, parsing or inference. It is more concerned with the organisation of processing resources than with the actual nature of the mechanisms involved.

The non-specificity (and non-formal character) of ICS means that mechanisms which determine how procedures are generated from buffered episodic memory are generated. It can be assumed that production composition, proceduralisation and tuning processes act to bring about learning. ICS does not seek to address the basic mechanistic properties of the cognitive systems. It is targetted at explaining the qualitative aspects of human information processing which underlie learning and processing difficulty such as often occur with computer users.

On the other hand, requirements for higher level explanation of information input and output processing seem to be satisfied by ICS. The more task oriented view of the characteristics of behaviour which reflect more generalized processing faculties (as described by Fodor 1983) are not accounted for in any detail.

The limitation of ICS is that, as a framework, it cannot do more than indicate the level at which the information input and output processing problems are likely to occur. In this sense it has achieved what it set out to do and, as Barnard admits, more detailed models are required to sort out the precise nature of processing and the way in which it is coordinated and how precisely learning is thought to occur, and what effects this has on performance.
Characterisation of the Model

Since ICS involves only one component; an architecture for human processing, as opposed to the multi component modelling approach of CCT, it is not surprising that it covers only a small part of the Usability Scoping Matrix. It does not account for features of the device or the type of UI which will affect users.

For the above reason, ICS does not tackle the principles of observability, consistency or retrievability. Nor does it address the evaluation factor system application. The model is further restricted in that the ICS architecture does not elucidate the more generalized aspects of processing involved in such activities as planning and organisation. This means that task related behaviour of users cannot be properly addressed, without extending the architecture to deal with higher level behavioural organisation such as plans, goals, schemas and so on, so that the principle of UCTDs and the evaluation factor target tasks are not dealt with.

UI type is dealt with to a certain extent. ICS tackles performance characteristics which would be altered by use of menus as opposed to commands. ICS model processes are also sensitive to input and required output features, which would be different depending upon the Type of UI being used. Other characteristics such as direct manipulation, multi-tasking and so on could be addressed because the framework is not constrained by performance assumptions based upon a limited set of cognitive and behavioural components (other process models are constrained by such assumptions, for example CCT would be unable to deal with multi-tasking because its processing parameters are based upon single contexts, ICS does not assume any parameters, so its non specificity gives it greater generality).

As an evaluation tool, then, the ICS framework can only tackle the qualitative aspects of performance relating to the principles of simplicity and compatibility. It can represent the complexity of the rules required to interpret incoming information to the cognitive system, and of the rules required to produce competent output. It also contains mapping procedures to predict mismatches between users' existing knowledge and the UI's presentation of output or expectations for input.

The explicit aim of Barnard is not to produce an account of some specific component of mental life such as short-term-memory, nor is it to produce predictions
<table>
<thead>
<tr>
<th>Evaluation Factors</th>
<th>Usability Principles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simplicity</td>
</tr>
<tr>
<td>Users (Us)</td>
<td>Yes</td>
</tr>
<tr>
<td>System Application (App)</td>
<td></td>
</tr>
<tr>
<td>User Interface (UI)</td>
<td>Yes</td>
</tr>
<tr>
<td>Target Tasks (TTs)</td>
<td>Yes</td>
</tr>
<tr>
<td>Acceptability of Performance (Acc)</td>
<td>Yes Qualitative</td>
</tr>
</tbody>
</table>
relating to performance under special and artificial circumstances. ICS is meant to be completely general; a basis for any kind of future model, be it for HCI or other applications. It is assumed that ICS should be integrated with or used to select other models for whatever application is appropriate.

The architecture's treatment of the human characteristics which underlie performance is meant to be generalizable, so that the principles of simplicity and consistency would be equally well dealt with by the model whatever the type of user, system application, UI type, and target tasks involved. ICS predicts what characteristics of an interaction language make it simple for users and compatible with their existing linguistic knowledge. It generates qualitative predictions about perception, cognition and action in learning and performance, unlike other process models which always assume special circumstances, such as training, or error free performance, and the model is clearly open to augmentation.

The matrix characterisation implies that ICS would have to rely on other models or specifications if it were to be applicable as a design or evaluation aid in its present state. This implication has, in fact, received some support since Barnard et al. (1986) built and evaluated an expert system design aid based upon ICS. They found that although the system generated useful predictions, it appeared to be hard to use without considerable psychological expertise. The problem was that the ES required a great deal of information requiring judgements based upon "implicit approximation" to be made by the system designer about certain characteristics of the device upon which to base the ICS predictions. The information requests were formulated in such a way that it was hard to see how a non psychologist would be able to give valid responses. An improvement would be to have the designer use a more straightforward device specification and augment the ICS model to include procedures which would enable it to derive its required input from that specification.

2.2.8 Command Language Grammar

The Command Language Grammar (CLG) is used to specify the dialogue component and screen layout of UIs from a top-down perspective. It grew out of early work on the GOMS modelling approach (Card, Moran & Newell 1976) which takes
a similar approach to the task of user modelling. CLG uses a task description to convey Goals, it describes both operators and methods but omits selection rules, though Moran admits these should be included.

CLG is based upon the linguistic view of system architectures of which Coutaz (1989) provides a useful review, where dialogue is separated from application enabling the interaction part of a system to deal, as it were with an abstract system, without the dialogue being committed to a particular view of the application software underlying the interface. This feature has interesting implications for its scope.

GOMS models can be described at different levels on the basis of what are taken to be the basic units of analysis. Similarly, CLG has a notion of levels such that it redescribes the interaction in terms of the tasks, semantics, syntax and interactions, each level aimed at highlighting different aspects of the structure of dialogue. The GOMS levels can be thought of as differing in the time-span of their average operators. However there are certain aspects of CLG which cannot be described in terms of timing (e.g. the Semantic Level). So that, in fact CLG and GOMS have rather different concepts in mind when they talk of levels. Indeed the term 'level' should always be used with caution when referring to user modelling. Finally, there is no conceptual model in GOMS, it is a pure performance model, only representing the knowledge required to execute tasks. CLG may be an improvement in this sense since, although it is strongly performance oriented, it does try to provide a limited semantic analysis to rationalise the performance.

Classification of the Model

CLG is a UI design technique (Moran actually refers to it as a "design representation" which can assist UI designers) which takes a top-down approach. In this sense it resembles functional systems design methods such as Yourdon and Constantine's Structured Design (1978) which rely throughout the design cycle on complete and increasingly detailed, specifications of the system. The early specifications are very high level with detail about possible implementations left unspecified this approach affords the designer freedom to organise general structure before going on to work out sub-structures functionality and implementation solutions.
In the same way CLG first focuses on what tasks the UI must support and then goes on to address how these tasks will be organised and in detail in the UI. However the focus of CLG means that implementation issues are not explicitly dealt with. The method focuses on UI functionality, but not on how that functionality will be supplied by the software.

The approach does not incorporate an explicit psychological model of the user, rather the structure of interaction being designed for the UI is modelled in a number of ways which would be salient to a user, reflecting the different ways and levels in which the author assumes human knowledge is organised. This makes CLG a psychologically sensitive interaction model of the UI which expresses features of a user representation of the interaction.

Figure 2.11
Components of CLG Description
and Levels Within Each Component

| Conceptual Component: | Task Level |
|                       | Semantic Level |
| Communication Component: | Syntactic Level |
|                        | Interaction Level |
| Physical Component:    | (Spatial Layout Level) |
|                        | (Device Level) |

CLG has no process model of the user, so it cannot be considered to be a performance model as can GOMS. On the other hand it can be seen as a competence grammar but one which focuses on the UI rather than the user. This may appear to be an odd concept, but Moran (1981) explicitly states that as a design tool CLG must be "extensive enough to cover all existing command languages, plus all conceivable extensions to them". In other words CLG aims to fulfil the role of a competence grammar which will constrain all possible interaction languages which a
designer may generate for the UI; hence its name Command Language Grammar.

Scope of the Model

CLG is notable for its exhaustive approach to the problem of tackling aspects of the structure of the interaction. As stated above, the approach requires a complete UI specification at every stage of the design cycle, starting with the most general and working down to the most detailed level. The system interface is described in terms of three distinct components each having two associated levels of description as shown in figure 2.11.

The levels of interest here are the Task, Semantic, Syntactic and Interaction levels. The last two levels are more concerned with the physical, ergonomic aspects of the interface, and are not fully described by Moran (1981). In this discussion, therefore, we focus on the psychological, cognitive and knowledge structuring aspects of the interface.

The purpose of the Level structure of CLG is to separate out the conceptual model of a system from its command language and to show the relationship between them.

Thus Moran hopes that, using this analysis, it should be a fairly straightforward task to generate a conceptual user model from task analysis specifications. The levels of analysis then map onto each other using a set of more or less explicit rules (see Moran 1981). In the case of mapping between the Syntactic Level and the Interaction Level these rules are quite formal and precise. The process of mapping from one level to another in sequence results in a non-random approach to the specification of the command language and dialogue conventions, with respect to human factors considerations.

In his paper Moran (1978) then goes on to give an example of the application of CLG to a "toy" message-file handling system called EG. Although the system itself is extremely simple, the description using CLG is very lengthy, as a result of its redescription at four different levels. This example is a good indication of the need for models that can be developed into computer supported toolkits. The amount of data generated by the analysis of even a very simple system is large. A
full-scale CLG analysis of a real, more complex system would require an amount of effort beyond the capabilities of unassisted human analysts. A brief overview of the steps in the procedure for applying a CLG approach to UI design is as follows.

1. Moran recommends that the best method of approach is to begin by writing informal descriptions initially and then to formalize the parts of the descriptions as necessary. He also states that symbols should be generated on demand as the description requires them rather than trying to anticipate their need.

2. CLG tackles formalization by categorising the elements at each level according to their nature as described in figure 2.12. The elements of interaction are specified using a LISP-like pseudo-code, the idea being that this encourages the analyst to specify things precisely and unambiguously. All levels are described using this code which Moran claims has formal specification powers. However the code is curious because, when describing operations and methods, it mixes both declarative and imperative statements (the former type generally preceding the latter).

*Entities* are the elements upon which the rules at each level of description operate, for example, at the task level a file is an entity, and at the semantic level, where the system with which the task is to be executed is considered, the system, messages, directories, etc are all entities. In addition, each level includes descriptions of the organised activities required to achieve certain goals or results. *Operations*, commands and rules are specified before describing the *methods*, which are comprised of sequences of specified operations, at each level (except the task level) for the purpose of specifying the meaning of symbols for more detailed action descriptions. The Task Level is not required to specify methods as these are not part of the task itself. Nor does the Task Level make any assumptions about the specific features of the system.

Examples of notated task elements relating to the "EG system" specification (Moran 1981) are as follows:
Figure 2.12
Levels of Description of CLG
with Elements at Each Level

<table>
<thead>
<tr>
<th>Task Level Description</th>
<th>Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tasks</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Semantic Level Description</th>
<th>Conceptual Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Conceptual Operations</td>
</tr>
<tr>
<td></td>
<td>Conceptual Methods</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Syntactic Level Description</th>
<th>Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Commands to Enter EG CONTEXT</td>
</tr>
<tr>
<td></td>
<td>Commands in EG CONTEXT</td>
</tr>
<tr>
<td></td>
<td>Methods</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interaction Level Description</th>
<th>Rules for Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rules for Arguments</td>
</tr>
<tr>
<td></td>
<td>Methods</td>
</tr>
</tbody>
</table>

An Entity

MESSAGE = (AN ENTITY

  REPRESENTS (A SEND-MESSAGE)
  NAME = "Message"
  AGE = (ONE-OF: OLD, NEW)

(* A MESSAGE has a Header and a Body.
  The Header contains the fields To, From, Date, Time,
  and Subject.
  The Body contains arbitrary text.)

(* The AGE is a time-dependent mark: see MAILBOX))
An Operation

DELETE = (A SYSTEM OPERATION
       OBJECT = (A PARAMETER
                  VALUE = (A MESSAGE))
       (* The OBJECT is removed from MAILBOX and its
       SUMMARY is removed from DIRECTORY

A Method (from the semantic level)

SEM-M4a = A SEMANTIC-METHOD
       FOR GET-INFORMATION
       DO (SEQ: (START EG-SYSTEM)
            (SHOW DIRECTORY)
            (LOOK AT DIRECTORY FOR (A MESSAGE))
            (SHOW (THE RESULT OF LOOK))
            (DELETE (THE RESULT OF LOOK))
            (STOP EG-SYSTEM)

In addition to the LISP-like specifications of entities operations and methods the
description makes use of hierarchies of tasks and of syntax to clarify the structure
of the procedures used in the interaction. Such hierarchies are rather like the goal
structures used in GOMS (Card et al 1983). Each node in the tree represents the
goal of some activity or operation, and the graph is processed bottom first left-to-
right. Also, the interaction constituents are composed into interaction trees in
order to describe interactions. Using these methods the interface of EG can be
examined from many perspectives, each on having the capacity to reveal inconsis-
tencies and complexity.

The rationale for having four levels of description is that the designer is being
encouraged to base the UI design around a user's conceptual model of the tasks (as
opposed to the designer's model). The initial task level specification is reformu-
lated through a number of explicit levels because the user's conceptual model is
assumed to be multi level also. Moran states that users organise knowledge about
tasks at a number of levels which may be general task structures, right down to
detailed representations of orders of key presses. They need knowledge at all these
levels before they can complete tasks using a device such as EG, so the UI must support and reflect this knowledge. The best way of ensuring that this is what happens is to maintain the user conceptual model right from the high-level early specification through to the detailed specification of the actual command-name and structure specification.

3. The purpose of each level of description is as follows:

The Task Level represents the purpose of a system by enumerating the tasks it is supposed to do. It is more than an abstract task analysis as it is dependent on the system and shows how the system fits into the task environment. Tasks are organised in hierarchies with base tasks being at the lowest level. Base tasks involve automatic short bursts of activity in general. Moran states that when the other levels refer back to the Task Level for a set of criterion tasks (which must include the whole task structure) the most likely level, in the task hierarchy, to be useful will be the base task level since more commonalities between tasks will be highlighted. Prior to this stage in the design a task allocation and task support analysis would be carried out.

The Semantic Level adds to the conceptual model that the user brings to the system those concepts embodied in the system. It defines, in an abstract way, the functional capability of the system and explains them in the simplest way possible. Once the concepts have been enumerated at this level, the analyst then has a set of semantic "primitives" that can be used to define the meaning of objects at the Syntactic Level. The purpose of the Semantic Level is to explain the functions of the system in as simple a way as possible. Complexity here refers to how closely the Semantic Level concepts are matched to the user's prior concepts (to determine complexity would require some empirical analysis of potential user's concepts).

The Syntactic Level is concerned with how the system operations get evoked. It focuses on the relationship between what the system does and what the user must do, or say to get it done. The purpose here is to examine the efficacy of the user's control of the system.

The elements at this level are, according to Moran, the building blocks from which all command languages can be built. Generative rules are applied to the
specification to produce valid interaction structures. The communication dialogue
between user and system is structured in such a way as to specify "what operators
can be evoked when (commands and contexts; contexts are used to capture the
kind of state dependent information represented in the GTNs used by Kieras and
Polson in CCT), when and how entities are to be referred to (arguments and
descriptors), what information will be remembered during the course of the dialo-
gue (state variables), and what information the system will make available to the
user and when (display areas). The Syntactic Level also specifies how the system
operations are to be packaged for the user”.

The syntactic level preserves the logic of the semantic methods and enables the
analyst to structure interactions, system responses and side effects in such a way as
to capitalise on the structure of the users tasks, for example if a particular action A
in a semantic method is always followed by another action B, the analyst can
specify a command at the syntactic level which elicits A then B, without the user
having to type in a second command.

The notion of command context is used at this level to represent implicit modality
of a system, and the effects and side effects of commands can also be captured.
Moran claims that these devices give CLG the ability to deal with device
behaviour and the effects of states. Unfortunately this claim is not substantiated
by any attempts at proving the variety and consistency of the specification.

**The Interaction Level** has the aim of identifying which procedures enable syntac-
tic elements to be interpreted by the the system but not the order in which the vari-
ous parts of commands are specified; this is done at the Syntactic Level. The basic
Interaction Level constituent is the *specification* of a syntactic element. The pre-
cise actions required to specify commands and arguments are identified.

The interaction level is structured in terms of the syntactic hierarchy which is a
tree made up of those syntactic elements that the user specifies to the system (con-
texts, commands, arguments and descriptors). Each element of the hierarchy is
redescribed in terms of how it will be specified and when it will be interpreted.
This process elaborates the syntactic hierarchy into an interaction tree.

Consistency and simplicity can be described at the syntactic and interaction levels
using the CLG notation to capture effects of commands and special rules to represent similar interactions. Command rules and argument rules which capture commonalities between various parts of the dialogue and impose relational ordering constraints upon the components of dialogues are used at the interaction level to clearly describe nature of the specific components at this level. This feature very much resembles the use of rewrite rules in the TAG notation. By using these rules the dialogue can be described in such a way that redundancy, complexity, and inconsistency should be easier to identify.

The main problem in CLG is translation from one level description down to the next or from one description within one level to another within that level (e.g. task entities to task structures). This is achieved using what Moran refers to as "mapping rules" which help to transform one level description down to the next; for example Task Level descriptions of tasks are mapped to the Semantic Level by redescribing them using the Semantic Level entities and operations to construct methods which will achieve the same goals. These rules help to ensure that the logic and structure of the CLG representation is consistent. Consistency is highly important to the design as it is one of the features of the UI which can improve the user's ability to predict its behaviour. Unfortunately, as Moran points out, the Task and Semantic Levels are themselves informal, so mappings between them cannot be precise, and there are only informal rules for dealing with the transformation from the Semantic to the Syntactic Level. This could mean that designers will have some difficulty in carrying out this mapping process successfully.

Moran states that CLG can be looked at from three angles:

The Linguistic View which sees CLG as an analysis of the structure of command language systems.

The Design View which sees CLG as a representation for different systems as they are being designed.

The Psychological View which sees CLG as a model of the different kinds of knowledge that users have about systems.

The linguistic view of CLG attempts to specify the structure inherent in all possible command languages for a given UI design. This requires that the specification
covers all existing command languages, as well as extensions to them, and that it rejects systems that cannot function as command languages.

Moran claims that CLG has advantages over state transition diagrams because it is more restrictive and will generate fewer different command languages. In essence it is the user conceptual model and the task orientedness of the approach that ensure that CLG is capable of avoiding the absurdities possible in state transition network specifications.

From the *psychological view*, many of the features of CLG are quite plausible, but they need to be validated. CLG suggests that the user's knowledge of the system is layered and highly redundant, and that this means that although the user may know what he or she wants to do and how to do it at certain levels, he or she may not have the knowledge required to do that task using a particular system (i.e. does not have the interaction level knowledge). This feature could be tested by getting users to describe how to do various tasks using various systems for which they have varying degrees of knowledge and observing their performance.

Another feature of CLG is that it predicts that systems requiring fewer rules to describe will be faster to learn and will result in fewer errors by the user. In more detail, similar rules will interfere with each other, the observable result of this will be that users will confuse similar rules and make errors of generalization. When rules are completely consistent generalization will be less error prone. Moran adds that CLG should allow time and other performance measure predictions to be made for tasks of varying difficulty. All that is required are some processing assumptions as in GOMS or CCT, however these are not explicitly incorporated in the model at present. This means that behaviour such as error handling and learning are not readily captured by the current CLG formalism.

The *design view* of CLG is that it forces the designer to create a conceptual model of the system to guide the design. Such a design may be easier for the user to integrate with existing knowledge than would a design which left the user to induce from, various fragmentary clues, what was going on. Moran (1981) gives various examples of how a well formed conceptual model presented to the user can improve understanding of the device behaviour and improve learning.
Moran himself provides a summary of his own perceptions of the limitations of this analysis, (for more details see Moran 1983). The criticisms which he makes which are of most interest here are brought out in the following discussion.

CLG is strongly concerned with how a user uses his or her knowledge. However Moran admits that this model is weak in the sense that there is no characterisation of the user's background knowledge, that is, any experience of the system, or techniques for learning about the system, are not considered by the model. This is, of course a reflection on the fact that general human knowledge is extremely difficult to describe. Further CLG shows how the user uses knowledge about the system to accomplish tasks, but does not show how to use this knowledge when the user makes an error or whether his or her knowledge is adequate for dealing with error situations. There is no control structure for using knowledge in CLG (except for its procedure interpreter which contains the rules which determine how procedures are implemented). This means that CLG requires extension before it can fully model the user, with respect to knowledge, from a psychological point of view.

Other criticisms of CLG to which Moran himself admits include the fact that CLG only describes concepts at the Semantic Level, it does not analyze them and therefore cannot tell us anything about their nature and how this will affect performance. Also there is no rationalisation of the procedures, particularly at the Task and Semantic Levels which prevents the analyst from determining whether a given procedure is easy or difficult for a user to understand.

There is no description of a Syntactic Task, which would look at how the user organises the task in terms of the system specific characteristics. This means that the true nature of the user's interaction efficiency cannot be brought out by the formalization. A similar point about the incompleteness of certain levels is that there is no conceptual model of the interaction process at the Interaction Level, which the user is sure to have. Finally, the use of rules in CLG should be extended beyond the Interaction Level because a basic goal of CLG is that the system's simplicity should be predicted by the simplicity of its description. Expression of the characteristics of the system at different levels in terms of rules would highlight redundancy of knowledge and inconsistencies in the task structure within and between levels.
Characterisation of the Model

CLG promises many things without actually demonstrating them. This explains the number of possible aspects addressed by the technique. The notion of a user's conceptual model of a system and its multi-level properties is not sufficient to enable the grammar to claim psychological validity if the contents of the multi-level representations and their structure are not also empirically based.

Although CLG makes explicit statements about the behaviour of the application as it affects (prompts or displays) or reacts (responds or interprets) to interaction, these statements are not presented within a coherent model of the system as a finite-state machine. In other words they represent an even less complete view of the application than do the Generalized Transition Networks of CCT. The selectivity which results means that the representation is probably inadequate for capturing the properties of the system application and its behaviour.

This may seem rather finicky as a criticism, but the notation is presented as having similar properties in terms of formal power to Augmented Transition Networks (which are similar to the Generalized Transition Networks described in Chapter 2 in the description of CCT). However the CLG notation is very poorly suited to dealing with state behaviour of a system because of its inherent multi-level nature which distributes information about contexts, actions, and effects between the syntactic and interaction levels, and as a model of the virtual machine it suffers from all of the shortcomings of the GTNs used by Kieras and Polson (1985) and criticised by Green et al (1988) in terms of selectiveness of what is explicitly modelled and what is not (i.e. what is not modelled could have important consequences in the implementation). However it is probably safe to say that CLG does address the more formal principles of usability with respect to the behaviour of the UI and the target tasks which are more directly captured by the notation.

CLG, like Reisner's Action Language, does not clearly elucidate psychological properties of task representations, but by the same token CLG is able to address the simplicity (if not from a truly psychologically real perspective) of the UI dialogue and target tasks. It clearly aims to capture UCTDs of the UI specification with respect to the UI and target tasks; in fact this is its main aim. Simplicity can be ascertained from a CLG specification by counting its rules, assessing their
### Table 2.9 Usability Scoping Matrix for C.I.G

<table>
<thead>
<tr>
<th>Evaluation Factors</th>
<th>Usability Principles</th>
<th>Accessibility</th>
<th>Consistency</th>
<th>Observability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users (Us)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Application (App)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Interface (UI)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Target Tasks (TTS)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acceptability (Acc)</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
confusability and naturalness, and assessing the overall size of the specification; more complex systems will involve larger numbers of command and argument rules as a proportion of the total size of the specification than will more simple ones. UCTDs are preserved by the CLG specification but would require some separate scientific analysis or empirical support from the analyst about the characteristics of prospective system users in order to derive an idea of their existing task knowledge with which the system should be compatible.

Moran claims that CLG will not generate absurd interaction languages, and that they will be appropriate to the meaning of the tasks with respect to the users and the required tasks. The inclusion of an explicit task level description of the conceptual model of the device is perhaps the central feature of CLG which aims to ensure that the system will indeed embody the principles of simplicity and UCTDs which non-user-oriented UI specifications of a purely formal character would fail to capture.

Compatibility is not addressed because the approach has no explicit component to deal with the content of users' knowledge representations. The appropriateness of one set of command names rather than another would not be explained by a CLG description.

Observability, consistency and retrievability are only partially dealt with by CLG if at all. Observability is captured with respect to the UI and TTs at the Semantic and Syntactic levels where the responses and activities of the system are described, including what it displays and where on the screen. Consistency is captured within the application by looking at the interpretations of commands at the Interaction Level, and for the interface and TTs by looking at the effects and side effects of commands at the Syntactic Level. Retrievability may possibly be captured by making assertions within Syntactic Level command and context descriptions; for example that the effects of each command within a context are reversible, that each command has an opposite, and that it does not invalidate commands which allow the user to exit or enter any other state.

However, the difficulty of using the unsupported CLG specification approach to capture all of these formal properties would require a great deal of work, and given the lack of an account for the underlying application behaviour, it is improbable
that the approach would ever be used for analysing these properties because the scope of the analysis would not extend beyond the states involved in error free performance. Were users to deviate from the methods outlined in the specification, the resulting application states could still prove to be catastrophic in terms of unrepresented but important information inconsistent effects of commands and inability to exit from particular states.

Strictly speaking, in its current form, CLG has not been proved to generate predictions about users performance which would be sufficient to determine whether a UI is acceptable. However, given its similarity to the tested GOMS formalisms, it may be safe to assume that certain performance predictions could be derived from the model for simplicity and UCTDs (Moran 1981). As discussed earlier, the Interaction Level specification can be used in much the same way as a GOMS model (at the keystroke level) as a basis for predicting performance times.

2.3 Summary

Eight HCI Design and Evaluative Techniques (HCI DETs) have been reviewed and characterised according to their ability to address usability of UIs with respect to certain important evaluation factors. It would appear that none of the techniques is capable of addressing all of the possible issues which might concern a designer and that there are certain areas which these techniques are unable to address. The most notable reason for this being the weak treatment given to device characteristics which could alter the value of an assessment of usability in an unpredictable way if not properly specified and accounted for by the technique.

There is also a major problem of the complexity of approaches themselves; their attempts to achieve formal power and predictive precision mean that, for some approaches, much effort needs to be invested in their construction. This factor means that for realistically applicable approaches there is a breadth depth tradeoff in power, otherwise the approaches risk becoming too difficult and unwieldy to use. Whether depth (accuracy and predictive precision) or breadth (generality) is more appropriate for a useful model may be a question for serious debate. Barnard (1987, p115) states "it is now ... becoming widely accepted that predictive needs for design purposes are best served by approximate models rather than by theories that are necessarily formally adequate". In other words he believes that the holism
of an evaluation or design approach is more important than its precision. Qualitative aspects of a device and its specification or user's model may have more impact on its ultimate usability than precise quantitative aspects.

Reisner (1982) raises two points about application of HCI DETs as tools for assisting designers. She states that the tool should make predictions about ease of use and that the tool itself must be easy to use. Many of the models discussed in this chapter satisfy the first requirement, but none of them seems to have been proved to be easy to use. Further questions are raised by the lack of evidence available on the applicability of these techniques to real design situations. We have seen what it is that these techniques aim to do for UI designers and analysts; their descriptive and predictive scope have been tackled together with how, in theory, they approach their aims. What we have not seen is how the techniques deal with real design problems and constraints, for example few of the techniques address the problem of how the results of evaluations would feed back into, and usefully alter the design of a UI. What are the design cycles envisaged by these models which would allow them to be effectively integrated. This is a question which has not been addressed explicitly by many of the authors.

The main aim of the following chapters in this thesis is not to question the theoretical bases for HCI DETs of the type described in this chapter. Instead, assuming that the claims for descriptive and predictive power are largely valid, we shall concentrate on how such techniques might actually be applied outside the research environment. In this area the literature appears to be particularly scant, and it is disturbing to contemplate that, for an applied science such as HCI surely is, there seem to be few case studies or demonstrations of HCI modelling techniques in actual UI design projects.

The next four chapters will address the nature of UI design in applied and commercial practice and will attempt to answer the following questions:

1. Are HCI DETs generally applied by UI designers?
2. Do HCI DETs address, or allow for some of the problems in design which could affect their implementation?
3. Are there certain conditions and practices in real design situations which can altogether prevent certain HCI DETs from being applied?
A further issue that will be addressed is whether there are requirements which UI designers have which have not, as yet been tackled by HCI DETs. It may be that some of the supposed design issues addressed by these techniques are of secondary importance in the real design situation, or that such issues are being superseded by newer issues with which HCI techniques have not yet caught up.
Chapter 3

HCI Techniques in UI Design and Evaluation: The Theorist's View and The Practitioner's View

3.1 Introduction

3.1.1 Overview

This chapter seeks to place the techniques reviewed in the previous chapter in some perspective with respect to their relevance to UI design and evaluation. In some senses it is a practical discussion of the implications raised by the nature of HCI DETs for their applicability which must be viewed as an important determinant of their validity aside from theoretical considerations.

The purpose of this chapter is to provide a wider view of user-oriented design and its implications for HCI DETs than was presented in the previous chapters. The design views of the techniques reviewed in Chapter 2 are considered and criticised. They are contrasted with other user-oriented views of design and with evidence provided by studies of HCI practice, or the lack of it in systems design. A number of questions are raised by the discrepancies between the design views of HCI techniques of the type reviewed in Chapter 2 and alternative design views.

3.1.2 Systems Design in General

The process of system design or development (development is viewed here as meaning a subset of design activities) frequently referred to as the software life-cycle (Sommerville 1989) can span a period of up to several years or more. This period covers the conception, building, implementation, adaptation, right up to obsolescence of the system. What the term software life-cycle may refer to could include a large number of diverse activities such as planning, market segmentation, requirements specification, scoping, modelling (of users, businesses, data, functions, systems), prototyping, solution generation, evaluation; which may include valida-
tion ("Are we building the right product?")}, verification ("Are we building the product right?")], and user testing ("alpha" and "beta" testing; initial controlled testing and in-situ testing respectively). Many of these will be iterated and used later in evolution of an existing system. They are but a few examples of activities that spring readily to mind.

The diversity and number of different software engineering activities means that it is almost impossible to provide a realistic definition of what is actually meant by the term system design. Many or all of the above activities may play a role in UI design. The manner in which software design of all parts of the system proceeds may be as varied as the people who carry it out. However there are certainly some commonalities between different design approaches, and this is probably more true of successful approaches. It is these successful commonalities which theoreticians attempt to embody as features in their methodologies and guidelines. The principles described in the preceding chapters are examples of such features. They represent UI design goals as opposed to design activities. However in this chapter we focus on UI design activities or 'practice', both ideal and observed.

Typically models of the software life-cycle are built, just as models of system users and machine behaviour are. Software life-cycle models are designed to clarify the problem of deciding what activities should be carried out and how they relate to one another in terms of precedence, inclusion, dependency and so on. One of the earliest such models was the "waterfall model" developed from Royce (1970). It is essentially a general specification of necessary, but not necessarily sufficient, design activities with a tentative structure which describes their sequential relationship in a design process. Figure 3.1 shows the waterfall model with possible iteration cycles; the name "waterfall" clearly comes from the sequential order of the stages.

This example is so trivial as to be almost meaningless as a model of a design method, however it serves to illustrate the bare minimum of a system design process which might result in a UI. What is important about this illustration is that it does not explicitly specify any activity related to ensuring that the system is really what HCI specialists might call a usable one. All that is strictly necessary for the system is that it is accepted (by someone) as a solution to some requirements specification.

This may seem a rather obscure point to make, but it is a sobering one; there is, in
many cases where life and limb are not at stake, no actual obligation on the part of a 
system designer to ensure that the UI really is satisfactory, beyond trying it out him 
or herself (to make sure it can be used). When the system is implemented, and 
users cannot easily carry out their tasks with the system, the blame is not necessarily 
laid at the designer's feet. It is possibly more likely that users will feel intimidated 
by the system than that they will complain that the system is poorly designed.

Figure 3.1
Waterfall Model of Software Life-Cycle:
A Minimal Outline

In this chapter two different views of UI design will be discussed. The first is em-
bodied in attempts at specifying idealised or improved design and evaluation ap-
proaches; these are referred to as the HCI Theorist's View. The second comes from 
evidence relating to what actually happens in UI design; the Practitioner's View.

3.2 Theoretical HCI Views of UI Design and Evaluation

The HCI theorist's view of UI design can be seen as an idealised process, or a plan 
which, if successfully implemented, should produce better interfaces than would 
have been possible without it. The kind of plan we are talking about here is one 
which specifies what activities should be carried out and how they should be organ-
ised in order to reach the target goals or specification for the UI.
In the following sub-sections alternative theoretical views will be examined; these are the design approaches assumed by the HCI DETs discussed in the previous chapter. These will be contrasted in the following section with other design approaches recommended by researchers for generating more usable interfaces (User-Oriented approaches).

The eight approaches to modelling systems and users, seven of these with a view to improving the usability of UIs, described in Chapter 2, may be taken either as pure research exercises or as directly applicable techniques. A clear distinction can be drawn between those which claim to be directly applicable to UI design and evaluation, and those which do not.

Some of the approaches reviewed in Chapter 2; BIMs (Morton et al 1979), Formal Grammar (Reisner 1981, 1982), ACT* (Anderson 1983), and ICS (Barnard 1985) are specified, by their authors as being research tools, or possibly, in the case of BIMs and ICS, as tools that could be used to decide between appropriate (but perhaps not applicable) evaluative approaches on the basis of their scope. That is not to say that these tools could not be used to improve UI design by some method, but that they do not attempt to meet the criterion of direct applicability.

On the other hand, TAG (Payne & Green 1986), GOMS (Card et al 1983), CCT (Kieras & Polson 1985) and CLG (Moran 1981) do make explicit claims about their applicability to UI design or evaluation, it is these claims that we will be addressing here. These claims can be illustrated by a series of quotes from the authors (my italics).

"Task-action grammars ... provide designers with an analytic tool for exposing the configural properties of task languages." (Payne & Green 1986; pp 93)

"Our implicit advice to the system designer has been to use these [GOMS] models in design."

"We have expressed some of the results in this book as a set of design principles to aid in the design of systems for human-computer interaction." (Card et al 1983; pp 417 & 424)

"[CCT] ...would permit the detailed evaluation of the relative complexity of alterna-
tive designs of a device"
"...the formal representations that have been represented in this paper provide the tools necessary to explore the psychological aspects of the complexity of a device, and provide the quantitative metrics for user complexity that are necessary for applications of these theoretical ideas for the design of actual products." (Kieras & Polson 1985; pps 392 & 393)

"CLG guides the designer by ordering the decisions he must make."
"Several...evaluation measures could be derived from CLG descriptions to guide the designer..." (Moran 1981; pps 45 & 47)

What these quotes demonstrate is that these techniques are meant to be applied by systems designers to their UIs. How they are meant to be applied is addressed with varying degrees of clarity. To begin with TAG does not appear to come with any firm view of how it might be applied in a real design situation. It's theoretical analytic aims and power are clearly described by the authors but the more practical issues are left to the reader to consider.

GOMS, CCT, and CLG on the other hand all come with fairly explicit assumptions about the nature of the design process, and the methods which will be involved in carrying out the analysis. A description of each set of assumptions will be given in the following.

3.2.1 The GOMS View of the Design Process

Procedural Aspects of Design

GOMS provides explicit advice for the designer attempting to apply the approach in practice. The authors present ten principles which should guide the designer towards a better UI:

1. Early in the design process consider the psychology of the user and the design of the user interface.

2. Specify the performance requirements.
3. Specify the user population.

4. Specify the tasks.

5. Specify the methods to do the tasks.

6. Match the method analysis to the level of commitment in the design process.

7. To reduce the performance time of a task by an expert, eliminate operators from the method for doing the task. This can be done at any level of analysis.

8. Design the set of alternative methods for a task so that the rule for selecting each alternative is clear to the user and easy to apply.

9. Design a set of error recovery methods.

10. Analyse the sensitivity of performance predictions to assumptions.

These principles of practice are aimed at improving the designers exploitation of the information which is required to design a usable interface. They also stipulate certain design activities which themselves should improve the probability that the UI will be satisfactory. These principles aim to foster good design practice, rather than characterising good design features.

The design process is viewed in GOMS as proceeding in a complex, iterative fashion in which various parts of the design are incrementally generated, evaluated and integrated. The process is seen as a set of different design functions each attending to a specific design subproblem:

\[ \text{Design Process} = \text{a set of Design Functions} \]

No attempt is made to describe a taxonomy of design functions but some examples are provided, and the functions are categorised by Card et al (1983) into structural, evaluative and parametric design functions.

1. Structural design is where the system, or system part, is actually configured or
restructured to satisfy requirements specifications. Examples of functions in this category would be; identification of an opportunity, problem diagnosis, improvement generation, and generation of a new structure.

2. Evaluation is where the system (or part) has been built or specified and requires an analysis of its performance, perhaps for the purpose of making improvements in performance. Examples of functions in this category would be, analytical modelling (such as TAG), user-system simulation (such as GOMS or CCT), and user-trials.

3. Parametric design refers to situations where the structure of the system (or part) has been fixed and quantitative performance parameters, or ranges of parameters, need to be defined as part of the final system performance specification. Examples of functions in this category would be performance modelling (such as GOMS or CCT) and performance trials.

GOMS is assumed to be applicable to evaluation and parametric design functions but not necessarily to structural design, although the authors suggest that "partial inversions of more formal performance models [e.g. GOMS] can ... be used (e.g. to diagnose the causes of performance deficits)" (Card et al p407). What is implied here is that GOMS performance models could be analysed to detect design problems in the specifications from which they have been derived, and indicate improvements which would be beneficial for performance. Performance is characterised by the authors in the following formula containing relevant types of structural variables:

\[ \text{Task} + \text{User} + \text{Computer} \rightarrow \text{System Performance} \]

The application of an appropriate predictive model such as GOMS is characterised:

\[ \text{Model} (\text{Task, User, Computer}) \rightarrow \text{Performance Prediction} \]

The structural variables of the human-computer system - the task, the user, and the computer determine its performance variables. In the formulae, "task" variables are
roughly equivalent to the system application (S App) factor in the evaluation formula in Chapter 1, referring to the nature of the tasks the system is designed to perform and support (i.e. its functionality), but in this case also covering the nature of the model used to describe that performance. "Computer" variables are roughly equivalent to the UI type factor (UI) in Chapter 1 referring to the style of interaction, conventions, and layout. "User" (Us) variables are, of course, the same. Card et al give examples of instances of such variables, such as, for task variables; text-editing or graphics, for user variables; cognitive style or perceptual-motor skill, and for computer variables; dialogue style or naming conventions. They also describe types of performance variable, such as basic performance metrics including functionality, learning, time and errors, also subjective measures, extreme conditions, and memory variables.

Scope of Design Issues Addressed

Card et al (1983) admit that the research they carried out left a great deal to be done with respect to investigating structural variables. For example only one type of task domain was investigated, users were crudely characterised, although they are known to be more complex than the MHP and related GOMS models are. Only a limited number of different types of UI were investigated for which the model was shown to be sensitive to their differences. With respect to evaluative metrics, the authors state that they have only concentrated on performance time, but that this is not necessarily the most important metric.

This limited scope of GOMS means that it would be unlikely to capture all, and would possibly miss the most important of the system design weaknesses if it were inverted to diagnose causes of performance deficits. Even applying the model the normal way for predicting performance times might turn out to be a secondary consideration for certain applications. Furthermore, the aspects of human behaviour which GOMS represents might, in themselves, be of lesser importance. If the application in question happens to be a training and explanation system then a model of error free performance would be of no help in predicting where learning difficulties and errors would occur. In brief the limited scope of GOMS could mean that it will only be applicable to a small subset of system designs.

A more detailed view of the scope of the GOMS formalism within the three design
functions from Card et al (1983) is shown in figure 3.2. The specific activities included for each function are in fact activities which were carried out with the GOMS approach to test its utility for a number of possible design or evaluation functions (structural, evaluation and parametric functions; see earlier discussion). The evaluation function of GOMS may be put to use in two ways; it may be used to predict that user performance on one system will be superior to performance on another, or it may be used to predict that some set of performance standards required of the system will be met. For parametric (specification) GOMS evaluations GOMS performance predictions can be based on the most acceptable (the fastest, the most representative performance, the poorest performance, etc) parameter values, and they can show how basic operator performance can relate to overall task performance of comparable tasks, and which parameters are likely to be the most reliable. For structural design GOMS analyses can highlight the aspect of the system which has to be improved, they might possibly indicate what the actual cause of user problems is, however Card et al state that it is less certain that GOMS analyses would support generation and synthesis of improvements.

If we accept the findings of Card et al, we can assume that this represents the experimentally verified design scope of the GOMS formalism in the laboratory. Evidence from actual systems design studies is lacking.

Pragmatic Design Issues

Apart from the procedural design view that GOMS has and its modelling scope, there are a number of more pragmatic considerations. GUMS is a complex approach which involves a cognitive model of the human processor, and a production-style task analysis. These components would appear to require a certain amount of expertise both to understand and apply, and for real system design projects a GOMS specification could turn out to be large and time consuming to produce, particularly for a non-psychologist.

GOMS is assumed to be applicable in a design situation where iteration will take place. However the way in which GOMS specifications used in applied design should be altered is not exemplified by its authors. Consequently it is hard to tell what kinds of alterations might be necessary and how much work making adaptions to GOMS model might involve. If changing one part of a specification means
changing many other parts that depend upon it, then the whole iterative process could be very time consuming.

**Figure 3.2**
Design Activities Undertaken Using the GOMS Approach

<table>
<thead>
<tr>
<th>DESIGN FUNCTION</th>
<th>Specific Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EVALUATION</strong></td>
<td></td>
</tr>
<tr>
<td>Compare systems</td>
<td>Compare on given performance variables</td>
</tr>
<tr>
<td>Evaluate system</td>
<td>Compare against some standard</td>
</tr>
<tr>
<td><strong>PARAMETRIC DESIGN</strong></td>
<td></td>
</tr>
<tr>
<td>Optimise Parameter</td>
<td>Find best value on given performance variables</td>
</tr>
<tr>
<td>Analyse sensitivity</td>
<td>Relate parameter value to performance</td>
</tr>
<tr>
<td><strong>STRUCTURAL DESIGN</strong></td>
<td></td>
</tr>
<tr>
<td>Identify opportunity</td>
<td>Find place where the system can be improved</td>
</tr>
<tr>
<td>Diagnose problem</td>
<td>Pinpoint structural component causing problem</td>
</tr>
<tr>
<td>Generate improvement</td>
<td>Find structural change</td>
</tr>
<tr>
<td>Synthesise structure</td>
<td>Create new structure</td>
</tr>
</tbody>
</table>
To compound the problems of the analyst's experience with psychological and HCI theory and techniques, there is the uncertainty of not knowing in advance the most appropriate level at which to describe and model tasks. Modelling at several levels would be very time consuming, but there is no guarantee that for any application a particular model will be the best. Card et al found that the unit task level was most accurate for predicting sequences of actions with text editors, but this could be due to features of editors or similar systems, rather than features of users or GOMS itself.

In addition GOMS requires that the analyst carries out some empirical studies to determine the skill, style and typical operator times for prospective system users. Skill is important to the GOMS formalism because it will dictate the kinds of production rules that users might be expected to apply. Style is important because it will dictate the selection rules applied to alternative methods. Operator times are crucial because they underpin the time estimates for the units of action at whatever level of description the model is applied.

**Synopsis**

No organisation is provided as to how to determine the relationships between the structural functions of the envisaged design process, for example the authors do not state whether parametric design is strictly dependent on, or subsequent to, structural design. One would think that the interdependency of these activities in a system life-cycle might be just as crucial to the design outcome as their actual nature.

GOMS does not take into account the possibility that analysts will not be able to use the approach as effectively and accurately as its authors. Nor does it provide guidance on how to determine appropriate levels of analysis a priori to applying them. Furthermore the design environment would have to be supportive of detailed empirical pilot studies from which to formulate a particular model for the design in question.

GOMS assumes, without justification from design practice, that time will be sufficient for its application, and that the specifications upon which to base the predictions will be available early enough in the system life-cycle to permit those predictions to have a useful impact upon the design. Unfortunately GOMS (like other
HCI techniques) has not been thoroughly tested in applied design projects. This means that such assumptions may be ill-founded. The approach has only been tested on text-editors in the laboratory, and even more seriously it has not been compared with other approaches in design which might be more effective.

3.2.2 The CCT View of the Design Process

Procedural Aspects of Design

An analyst using CCT to evaluate a design must be able to build a GTN device representation as well as a GOMS style user's job-task representation. This would imply that CCT will be more time-consuming and difficult to apply than GOMS because more modelling work would have to be done. As with GOMS the system in question must already be specified to the extent that its behaviour can reasonably be simulated to a degree that will not seriously invalidate the model's predictions.

Polson (1987) provides a summary of the design process into which he believes CCT will be integrated. The top-down, stepwise-refinement design methodology assumed is not presented as being typical of all top-down approaches. It differs from traditional approaches in that it includes concurrent development of both the hardware and the software specifications as well as the user's representation of tasks performed by the system. It also involves iteration of the complete design process. As Gould & Lewis (1985) have demonstrated, several iterations of each of the stages involved in any design process will be necessary to develop a highly usable interface.

The four phases of the design process envisaged by CCT are

1. **System Definition Phase:** Specify the functional requirements of the system.

2. **Task Analysis Phase:** Specify the user's decomposition of each task performed by the system.

3. **Detailed Design Phase:** Specify the details of the user interface, including methods, menus, commands, etc.
4. Evaluation Phase: Evaluate the design using simulation methods.

The System Definition Phase is viewed as the specification of user requirements, implementation environment, basic UI structure, and system functionality. CCT assumes that these activities will all be dealt with by appropriate methods which are not described further by the authors.

The Task Analysis Phase involves specification of the top- and middle-levels of a user's goal structure for each task identified in the previous phase. Polson states that the levels are defined by a user's decomposition of a task into subtasks and, if necessary, sub-subtasks. The design team then develops the methods which will efficiently accomplish these tasks. The method suggests that specialists be engaged to conduct the task analysis which would presumably take place separately from the rest of the design process. No firm guidelines are given as to what actually characterises a top- or middle-level goal, or indeed a task. It must be assumed therefore that CCT relies on expertise of specialists in order to determine appropriate levels of description for the purpose of modelling the user's job-task knowledge.

The Detailed Design Phase includes the completion of the detailed specifications of the UI and sequences of user actions necessary for accomplishing each task from the previous phase. The design team must also develop a test suite. This specifies a set of tasks and the characteristics of each task to be involved in the usability evaluation of the design. The test suite must exercise each method necessary for completing the various tasks specified during the system definition phase. This process will inevitably demand valuable time and resources.

The Evaluation Phase covers the writing of the system model (using GTNs) and the production system models for each task. These tasks require considerable care and expertise with the representation methods. It is no easy task to generate a GTN of a complex UI, and specification of the production rules. If the production modelling is done without consideration of empirically based psychological theory relating to how humans organise knowledge about tasks, it may yield an arbitrary and unrepresentative model of the user. Bennett et al (1987) propose a special support tool based around a task modelling language (TML) which can be used to describe tasks and then translated by computer into production rules. The problem with this is that the tool is not generalizable to any set of tasks, and the degree to which an
analyst's style influences the nature of the TML specification makes it necessary that the same person carry out all TML descriptions (which could hold a project up significantly).

The design team then simulates execution of all the tasks in the test suite, deriving static and dynamic measures from the production system model for each task. These measures can then be used to formulate predictions about training time and productivity. The results can then be used to evaluate the design and plan changes, which will be implemented on the next iteration of the system design. Polson considers that this level of specificity of the design can be achieved in the early stages of the process of system design. He states "Cognitive Complexity Theory provides methods for generating quantitative predictions for training time and productivity early in the design cycle." (Polson, 1987, pp 228; my italics again). However, the degree to which this can actually work in practice has not been demonstrated.

CCT makes stringent demands upon the kind of design process it may be applied in. The assumption is that the facilities will exist; to make precise early specifications of detailed UI functionality; to integrate a LISP based simulation tool with a state-transition network specification of the device (i.e. requiring that a valid state-transition network representing the UI software be available, which might have to be supplied by a tool such as Wasserman's Rapid-USE (Wasserman, 1985; Wasserman et al, 1986) to be available). There is a further assumption that it will be possible to determine certain user performance parameters from which to generate the full predictive model. Given these claims that CCT represents an early evaluation technique, the design cycle described by Polson for application of CCT is not empirically demonstrated, it may not be representative and may not be integratable with certain systems analysis and design techniques. At worst, in practice CCT may only be applicable as a late evaluation technique in certain situations.

Bennett et al (1987) claim that, with the TML program supporting production generation, CCT is capable of assisting a development team in their decisions about which of two competing designs to choose. However they based most of their work on existing systems (table drawing tasks on two interactive document formatters, one being WYSIWYG and the other not). The suggestions they came up with on the basis of their analysis, for a new table formatter design were not clearly superior to subjective judgements, and their improved design was largely based upon the use
of defaults which severely restricted its flexibility and would have meant that it could only be used for a narrow range of circumstances.

**Scope of Design Issues Addressed**

Poison proposes that UI style does not significantly influence usability quoting research carried out by Whiteside et al (1985) as support. However this support is based upon a very limited file manipulation "benchmark task" designed by Magers (1983) to compare help systems. This hardly constitutes strong support for the idea that "the details of a given design, not the interface style, determine a system's usability". On the other hand Card et al (1983) demonstrated that performance using a mouse was superior to joystick or cursor-control keys, and Foley et al (1987) state that functionally equivalent designs may vary in their usability.

CCT does not concern itself too heavily with stylistic aspects of the device which cannot be captured by GTNs. Different types of UI may actually lead to different predictions if the GOMS model constructed is sensitive to alternative physical actions required by different UIs to accomplish a particular task. In other words, CCT may actually be capable of predicting that one UI type or style may be superior, in terms of user performance, to another. The claim that different styles of UI do not influence usability is not necessary to justify the utility of CCT and may require more empirical support than has been given, or a very restricted definition of the meaning of the term "style" as opposed to "details of design".

On the other hand, CCT like GOMS is not capable of modelling "look and feel" aspects of the UI or predicting that these can strongly influence user performance. This may be the reason why UI styles as a factor are not considered by Polson as important. By ignoring such aspects of the system in the simulations, and in validation studies of these simulations, it is easy to deny their importance for the predictive power of an evaluative approach.

There are a number of other reservations one would wish to make about the approach. First, as Wilson et al (1988) point out, the assumptions made about the performance characteristics of naive and expert users may require a considerable degree of refinement if the model is to be truly generalizable for application in design. Working memory in the production systems architecture, for example, is
not subject to decay which would significantly affect the predictions about naive
users whom the model describes as being heavily dependent upon conscious
appraisal of feedback from each step they take.

Also the style of different users (the probability of their selecting one out of a
number of possible procedures as modelled in GOMS) may not be captured well by
the model without a great deal of empirical study before building simulations. This
would be extremely time consuming, and in a real design situation the simulation
implementor might be tempted to write the rules, and to assign probabilities, on a
purely intuitive basis, thus losing the representative value of the model.

There are some significant problems with the how-it-works representation of the
device. The decisions made to nest certain aspects of the GTN simulation because
the analyst intuitively feels they do not impinge on the user are essentially arbitrary
and may be ill founded. An explicit procedure is required for deciding what aspects
are and are not nested in the device model. The importance of this criticism relates
to the fact that alternative, but equally valid device representations could lead to dif-
ferent predictions from the user processing simulation because it would be respond-
ing to different sets of system states.

With respect to the design potential of CCT, GTNs are only abstract representations
of the system's UI states. They do not provide any information about how the sys-
tem looks to the user (are menus vertical or horizontal, are icons distinguishable and
so on). Also the approach does not really encourage the designer to improve the
device with respect to complexity, for example the IBM Displaywriter editor's
structures for selecting edit functions are highly inefficient since they expect the
user to select the mode (e.g. DELETE, MOVE etc) before selecting string length,
the word, or the character to be edited. This modality is a poor design feature and
requires a more complex set of task representations (text selection sub-tasks for
every edit task) than if the edit verb came after the text selection (perhaps a general
edit text selection procedure, and single-mouse-click-or-key-press tasks for all edit
functions), which is now more common in modern UIs. Kieras & Polson (1985)
modelled the editing features of the Displaywriter with CCT but the analysis failed
to identify this problem. So although CCT claims to represent complexity, it may
no be a very useful tool for suggesting how to reduce it.
One of the most interesting characteristics of CCT is that it is proposed that the representation of the device should be modular "so that as much or as little of the device can be represented as desired" (Kieras & Polson 1985). This modular aspect means that the problems of complexity of the model itself can be partly avoided because the specification and simulation problems can be broken down into smaller simpler parts. One of the main problems with HCI modelling techniques is their complexity which makes them difficult to use and understand. The idea of imposing modularity on the device model in the simulation is a useful one permitting designers to evaluate only new or unusual features of a UI rather than the whole, which could save a great deal of time and effort.

The approach has not been tested on UIs in general. Most of the reported research was carried out on text editors or formatters (e.g. Kieras & Polson, 1985; Polson, 1987; Bennett et al 1987) The special nature of text-editors which are relatively straightforward to characterise using formal descriptions, and which are almost all UI oriented with relatively little other functionality, means that they are not a good testing ground for approaches which seek to be generalizable. This criticism is one which can probably be applied to most of the HCI modelling techniques, but is particularly relevant here because CCT does rely on formalizable specifications for the simulations and attempts to model the virtual machine states without addressing certain aspects of its behaviour. Text-editors therefore happen to be the most suitable, large group of systems which fit the requirements of CCT. Other system UIs may turn out to be extremely difficult to evaluate with CCT, for example Knowles (1988) provides evidence which suggests that CCT is very poor at capturing various qualities of certain types of UI (namely CAD tools) which strongly influence their complexity.

CCT like GOMS is a reactive rather than a generative design tool, in that it does not drive the design decisions which are to be evaluated. During the task analysis phase no guidance is given by the approach as to how to select efficient methods for task execution. Without some explicit design principles such as ensuring reusability of UI components to maintain simplicity, designers may have to go through many, expensive iterations before a highly usable interface results. This criticism follows through to the production system modelling and GTN modelling which the approach requires. Few clear guidelines are given as to how to determine the psychological validity of a production system, or how to determine what aspects of
the system to represent in the GTN. This means that analysts without appropriate skills risk generating unrepresentative and inaccurate models.

**Pragmatic Design Issues**

Kieras and Polson (1985) state that the "device and job-task formalisms... have been chosen for their ease of use in the computer simulation techniques..." Unfortunately this ease of use does not seem to have been evaluated for complexity as the approach suggests UIs ought to be. This seems to be a rather cynical comment to make, but the authors are making an explicit claim about the nature of CCT which should be evaluated by user-testing in the same way that a UI might be.

Even given the unsubstantiated claim that CCT is easy to use, Polson suggests that specialists be allocated the task of specifying tasks and deciding the grain of analysis of the simulations. As there are no clear principles for determining in advance what an appropriate grain of analysis might be, it is not clear exactly what kind of expertise is required by the specialists undertaking such a task.

Card et al had the same problem in being unable to determine in advance, what level might be the most accurate for the task of text-editing. They had to determine this by implementing different models, at several levels of analysis, of one user carrying out one set of tasks on one application. There are no guarantees that the best level will not vary unpredictably for each different set of users, tasks and applications which might be simulated. What is missing is a more explicit statement of the factors which might determine the level at which performance might be most accurately simulated. For example if the low level components of interaction vary widely in a direct manipulation UI, they might not in a command driven UI. CCT would not supply any method for ascertaining that modelling basic operations of the direct manipulation UI would yield far less accurate predictions than modelling the command driven one would.

Another pragmatic difficulty of CCT is that it could add considerably to the time taken to pass through the four design phases, but since it cannot drive design itself, it may not reduce the total time taken to reach a state where the UI is actually usable. A stringent requirement of this and any other technique is that the diagnostic utility of the approach (i.e. the time and cost that is saved by the simulation’s
identification of problem areas and performance predictions) must outweigh the
time and cost of building the simulations in the first place.

This requirement might not be so important if there were no other methods for diag-
nosing sources of potential user difficulty, however CCT will be competing in
applied design projects with alternative evaluation methods such as user testing of
prototypes, UI demonstrations and mock ups. These may capture aspects of the UI
influencing usability that CCT would miss, such as compatibility with user's expec-
tations. If these methods provide more information, are cheaper, and can be applied
earlier on then they will be selected in preference to CCT.

Synopsis

Although the CCT view of design does impose a procedural organisation on the
process of development and explicitly states that iteration will be necessary. It is
vague about the precise nature of the procedures which will yield the required infor-
mation for the modelling and also about the way in which solutions to problems
might be generated from the model.

It places demands for expertise in fields of task analysis, production systems model-
ing, and GTN specification which would be difficult to satisfy with a small design
team. Like GOMS it does not account for the possibility that analysts other than the
authors might be unable to construct such accurate simulations of real user perfor-
ance. Nor does it address what advantages it might have over alternative methods
of evaluation in terms of overall time and cost savings in the design process.

It is assumed that the time taken to build and run the simulations and modify them
for each iteration will not be prohibitive, and that the advantages gained will
outweigh the investment. However, it is clear that CCT leaves certain issues, such
as style of interaction, and compatibility unaddressed. This would mean that a truly
conscientious, well equipped design team (such as might apply CCT) would prob-
ably have to conduct user trials to determine the influence of these issues as early as
possible, perhaps using mock ups or prototypes. CCT must therefore prove itself
capable of addressing issues which competing methods cannot, before it will be
viewed as a favoured design tool.
A final and important point is that CCT, like GOMS, has only been tested on text-editing systems. Although such systems are common, approaches which work with them may not be generalizable to other common systems.

3.2.3 The CLG View of the Design Process

Procedural Aspects of Design

CLG (Moran 1981) takes the view that a system UI can be designed by creating increasingly specific descriptions of it. It is assumed that the first abstract description captures the entire system's roles, but in such a way that the proposed user's conceptual model at this level can be specified at the next level. This user's conceptual model (UCM) will then be used to constrain the design throughout, the idea being that systems designed to reflect the UCM will be easier to reason about and to learn to use.

CLG does not claim to be a design methodology in that it does not specify the activities which will bring about the required CLG representations. However it is clear that for each level to be accomplished according to the model, certain goals must be achieved by the designer.

The Task Level requires that the following goals are satisfied:

A system scoping where tasks to be addressed by the system are specified.

A task analysis in which the designer must decide what parts of each task the system will support. The decisions must be based upon properties of the user and of the system.

The Semantic Level involves:

Deciding on a set of conceptual entities and operations which will cover all of the target tasks.

Ensuring that simplicity is achieved by using familiar concepts, as few concepts as possible, and methods which are as simple as possible.
The Syntactic Level involves:

- Deciding how to package conceptual operations into commands; whether commands should be simple, or complex and functionally extensive, should they be specific or general, multi-parameter or parameterless, and so on. Simplicity must be maintained above all.
- Deciding upon other syntactic devices such as command context structure, notations for designating objects, defaults, state variables, and command side-effects.

The Interaction Level is the level at which:

- Loose ends are tied up from the Syntactic Level; the designer decides how syntactic elements are ordered, and when they are interpreted. The aim is to produce an efficient, consistent and mnemonic set of interaction conventions for the user.

The Spatial Layout Level:

- Describes the arrangement of the input/output devices and the display graphics.

The Device Level:

- Describes all the remaining physical features of the UI.

Moran (1978, 1981) provides very little detail on the precise character of the last two levels. Therefore, these levels will not be considered further.

Within and between each level, CLG aims to guide the designer in maintaining the specification with mapping rules (guidelines or precise formalisms which enable one description to be transformed in an unambiguous fashion into another) in order to avoid inconsistency in the logic and structure of the design representation. The mapping rules should enable the designer to transform one description into an isomorphic description taking another view (within levels) or using another representation between levels. In this way tasks can be mapped to Semantic Level methods,
Syntactic Level Entities and Commands can be mapped to Syntactic Methods, and so on. However many of these mapping rules, particularly between the Task, Semantic, and Syntactic Levels are poorly defined, and will require skill and intuition to determine.

CLG does not provide clear procedural guidelines as to how the designer should gather the information required to compose a valid user's conceptual model and multi-level specification. At the Interaction Level rules rather like those of Reisner's Formal Grammar and Payne and Green's TAG are used to express simplicity and consistency, but at the other levels such rules are not included so that the designer has no guidance with respect to the embodiment of these principles in the design.

Further problems seem likely because there are no clues as to how to determine the appropriate level at which to describe entities, and the designer is not given help in determining what amount of detail is appropriate at any given level.

CLG provides a framework for ordering design decisions and "stratifying them into Levels". This means that the conceptual model of the system precedes the design of the interaction language. However the issue of what to do if the initial model leads to a final system that is somehow unworkable is not well addressed by CLG. This is because the methodology does not solve any of the software specification problems which tend to lead to early design decisions rapidly becoming fixed, and later, better informed decisions having little impact.

So CLG imposes a top-down step-wise refinement approach on the design of the UI which is regarded as being specifiable independently of the rest of the system. However no indication is given of how the UI specification might be integrated, and at what stage, with the rest of the system design.

Moran claims that the top-down approach to specification has the advantage of enabling each level to be evaluated before the designer goes onto the next. For the higher levels it may be difficult to determine what exactly represent good and bad features. Some features are suggested by Moran including those which represent deviations from the user conceptual model.
As for evaluation of the design as it proceeds, Moran proposes a number of possible metrics which could be applied to a CLG specification. Unfortunately these rely to a great extent on the system's already having been specified to the Syntactic or Interaction Level. Efficiency or speed of interactions could be judged from adding up the times of the number of physical actions users would have to perform in order to accomplish tasks. The Syntactic Level description can be evaluated for its optimality or redundancy. Syntactic methods would give an indication of the user's memory load during tasks. The overall length of the CLG description might be an indicator of the difficulty with which the system might be learnt by its users.

CLG descriptions cannot be user-tested until the final specification is available because the representations generated are not designed to be approved by users. Even simulations of the nature of CCT will be impossible in the Task, Semantic and Syntactic Levels because detailed information about the actual behaviour of the system is not available.

**Scope of Design Issues Addressed**

CLG addresses the conceptual model which a user might have of a system required to support a set of tasks. It is designed to provide a grammatical framework for specifying a satisfactory command language for the functions which will be carried out by the user with the system. The author claims that it provides a greater degree of restriction for the designer than do state transition networks because it avoids absurdity.

The main advantage that the approach is meant to provide is that it is intended to support attempts to model the user's knowledge of the system by forcing an explicit task representation to be provided, which may be based upon a task analysis. In doing so it constrains the designer to provide a system which preserved the features of the users existing or idealised task domain in the behaviour of the specified system and its UI. For example, in a well worked CLG design, the system will have a multi-layered structure which can be exploited by its users for the purpose of generalisation of rules or reminding of things which have been forgotten. The specification will represent the meanings of system commands as Semantic Level procedures which map well onto the user's conceptual model. Fewer rules will be designed into the system at the Interaction Level, so users will have less to learn.
Moran outlines a number of ways in which the CLG specification might be evaluated before implementation. The Task Level which expresses the scope of the system can be considered open to debate by interested parties, for example users, management and the system design team might all discuss the appropriateness of the task specification. The Semantic Level can be analysed with respect to the possible difficulty that users might have grasping the concepts involved. However at these higher levels there are no concrete methods as yet for evaluation.

A number of possible later evaluation metrics are proposed by Moran, given that the Syntactic and Interaction Levels have much in common with GOMS specifications (Moran 1981). Metrics could include speed of tasks, optimality of the language syntax, potential user memory load and overall complexity of the system (i.e. the size of its specification). Unfortunately the approach does not easily predict error sites as the user's cognitive processing characteristics (such as memory decay) are not explicitly captured in the descriptions. The same can be said of its ability to give a clear picture of what will and will not be easy to learn, beyond a simple rule count indicating general system complexity. Moran does not discuss what might be done to ensure that the metrics which CLG does supply are available early enough for designers to benefit from them. Given that evaluation of the early specifications (Task and Semantic Levels) is not well guided, CLG may only allow for late evaluation which may provide results which cannot be exploited by the design. Furthermore, the derivation of evaluation metrics from CLG specifications has not been demonstrated clearly as it has for GOMS and CCT, this makes it all the more difficult for another analyst to attempt to do so.

Another issue which needs to be dealt with is that of responding to requirements for change should some part of the system appear to be unsatisfactory. Moran admits that iteration is an unavoidable process in design but does not clarify how problems which emerge at one level might be traced back to earlier specifications. He does describe certain features which might indicate user problems as above, but does not clearly describe how these are to be dealt with during iteration from late detailed specifications going back to early high level specifications.
**Pragmatic Design Issues**

There is some worrying evidence (Sharratt 1987) to suggest that CLG may be rather difficult and time consuming to use. The main problems seem to stem from the application of mapping and consistency checking rules which were hard to apply correctly. Consequently the subjects in the study described made many errors such as missing mappings between related sets of entities, operations or methods. A further problem was that errors were easily propagated through to the next level in the specification. The effect of the difficulty experienced by the subjects in Sharratt's study was that they concentrated more attention on the complexities of CLG than they did on the design itself.

The very nature of CLG; its multi-level character, makes it a highly repetitive and time consuming method to apply. Considering what might actually take place in a design scenario involving CLG, it is likely that a complete CLG description would have to be generated before the first UI iteration could be produced. If this, for whatever reasons, turns out to be unacceptable then the whole process, or a significant part of it might have to be repeated in order to maintain the integrity of the user conceptual model.

Producing a CLG specification probably requires considerable painstaking care and effort because it has inherited the problem of confusing nested brackets from LISP which it resembles in notation. The nature of its notation and the concepts it conveys make it difficult to understand, and therefore not amenable to discussion with potential users or individuals who do not have the time to familiarise themselves with it. On the other hand, since it is not a simulation tool, the only way it can convey information is through being read - very carefully.

The approach assumes other unspecified methodologies will define user requirements, system scope and device behaviour which the designer would need to incorporate into the descriptions. Moran also assumes that the Syntactic and Interaction Level descriptions can dictate the ultimate syntax and interaction behaviour of the device, however systems are never so amenable that the designer has complete freedom in this respect. This problem is likely to be greatly exacerbated by the recent introduction of such tools as UIMSs, applications generators, rapid prototyping tools which exist in great numbers. Although these packages save time in
generating UIs, they can considerably constrain the freedom of the designer to build the system which ever way he or she sees fit. There will be parts of the developing UI which require that the CLG descriptions are violated. This possibility and how to deal with it is not accounted for by the approach.

There are a number of uncertainties within the approach such as the familiar problem of its not clarifying exactly what determines what level something ought to be described at. The Task and Semantic Levels are extremely informal and will probably demand psychological expertise from the designer. There is no clear method for mapping from the Informal Semantic Level to the more formal Syntactic Level. There is also uncertainty due to the lack of rules which could be used to represent knowledge at levels above the Interaction Level. All of these problems mean that a designer applying the technique might waste more time than the approach saved simply trying to understand or guess what the best way to tackle these issues might be.

Synopsis

CLG suffers from a number of problems which suggest that it might be extremely difficult to apply without the right background and expertise in HCI and psychology. The repetition of specifications in more and more detail will necessarily be more time consuming than an approach which only requires one specification to be generated. Even the very simple example EG Mail System description is very lengthy, real applications would be exponentially more complex to describe.

The other methods which will supply the information CLG uses are not described, and some of the methods within CLG itself are left unclarified, such as mapping from the Semantic to the Syntactic Level. This means that it is difficult to be certain that the technique is being applied properly even when the designer has grasped most of the concepts involved.

How the approach fits in with other design methods for the rest of the system is not easy to guess. The specification of the interaction characteristics is described without consideration of the possibility that the designer might not be able to find or develop a system flexible enough to permit their implementation. The metrics which could indicate potential design faults could emerge too late in the design to
prove useful. By the time the metrics were provided, it might be possible and safer for the non-HCI experienced designer to test the system on real users.

Apart from the many problems it appears to have, the sentiment behind CLG seems to be a laudable one in that it seeks to constrain the design space to solutions which will result in more usable interfaces. In other words the approach is intended to reduce the requirement for alterations to designs by guiding the designer in the right direction. This may partially excuse the approach's weakness with respect to dealing with evaluation and iteration. The approach is clearly in need of a set of guidelines which would help the designer to test the system on real users. Moran (1978, 1981) originally intended that such guidelines (design principles, design operations and design rules) should be supplied, but as yet they have not been made explicit.

3.2.4 Overview of HCI DETs Design Views

The three approaches described above embody theoretical views of HCI design, in that their views are not based upon application of the approaches in real design projects. The GOMS view is *procedural*, in that it provides design activity principles and describes the functions of each of the design activities it expects, and the specification's ability to fulfil these functions is tested in the laboratory. Despite its detailed account of how GOMS can be applied to design and evaluation, the *organisational* structure of a design approach is not accounted for. The order of, and relationships between, the various activities are not clarified.

CCT provides a highly structured account of the design process admitting sequence and iteration which contrasts with the GOMS approach. However, both GOMS and CCT are weak on how exactly the designer applies their methods, particularly when it comes to defining appropriate levels of description for tasks and writing production systems which are psychologically representative of users. CCT is also imprecise in determining what aspects of the device are to be nested in the GTNs. Furthermore CCT unlike GOMS and CLG does not account for generative support for UI designers. It is implied that mismatches between user- and device-representations indicate a need for redesign or training material but does not provide an explicit method for reaching solutions. Notably Kieras and Polson suggest that the
approach is an early evaluation tool but do not clearly support this claim.

CLG also provides a structured account of iterative design but like CCT fails to be specific about many crucial activities which the designer would be required to carry out. In spite of admitting iteration as a necessary part of design, CLG does not clarify how the approach deals with it, for example, if a problem is noticed at the Interaction Level, how can it be traced back, and what are the implications for the rest of the specification. Although CLG covers early design, it does not suggest any real metrics for evaluation until the later, detailed phases; for this reason it could be impotent in evaluation.

There are a number of characteristics shared by the three approaches discussed above (these characteristics apply to most of the approaches discussed in Chapter 2) which could prove prohibitive to their successful application in real design projects.

The approaches are complex and time consuming to apply: they require the analyst to re-specify the design in a way which is purely oriented to its usability (not its logic, reliability, efficiency, or security). Because there are other system requirements apart from ease of use, designers may not consider the effort of applying an unfamiliar, difficult, and unproven methodology is worth the risk of missing deadlines or running over budget.

Particularly important in view of the complexity of these approaches is the fact that none of them appears to have been evaluated by its authors for usability, nor for reliability when applied by non-HCI experts. Sharratt (1987) has confirmed this view with respect to CLG by demonstrating that MSc. students using the technique had considerable difficulty with the mapping rules, and maintaining consistency between the multiple representations, and that the use of CLG shifted their attention away from the experimental design problem they were given.

The approaches provide abstract specifications or behavioural representations which do not convey the look and feel of the system to the designer or user. Although, in some senses, they are designed as substitutes for informal user evaluations, they lose out on potentially valuable real-user input. This is because they are written in such a way that users would be unlikely to be able to contribute to the design specification before a prototype was available. The notations which describe
the system and interactive tasks are not familiar or simple and may prove quite
cryptic to the inexperienced.

The value of user input to design is considered by many (Glasson 1984, Jorgensen
1984, Gould & Lewis 1985, and others) to be enormous. Many of the benefits pro-
vided by this involvement of users are not diminished by use of HCI DETs because
the kind of input users can provide, although less scientifically precise, may be
much richer. It seems to be a waste of an opportunity to specify tasks and device
behaviour in such a way that users cannot comment upon it, although CCT does
claim, albeit without substantiation, that the device representations they provide are
comprehensible to users.

All of the multi-level specification approaches seem to be unclear about the pre-
cise nature of the levels of description they provide. The problem may be partly
a factor of what the levels represent to the analyst. The levels may be reflections of
human cognitive organisation which cannot be easily proved to be valid, or they
may be representative of different levels of detail with which a system may be
specified.

The essence of this argument is that most of the approaches which describe
representations as being multi-level take the latter position, with the possible excep-
tion of ICS (Barnard 1986), in that they do not rigorously validate the structure of
their representations. For this reason, it is never clear what features of tasks, con-
cepts, operations, and so on, make them suitable for description at any particular
level.

Payne & Green (1986) refer to routineness of simple tasks (terminal actions in an
analysis) as being dependent upon the user's level of skill, which is difficult for the
designer to predict. GOMS merely uses operator times to distinguish levels of
descriptions. This seems particularly inappropriate coming from a model which
acknowledges that time taken to accomplish some simple actions such as pointing a
mouse is heavily dependent on the characteristics of that task. For example Fitts
law (1954) states that mean movement time is a function of the log of distance \(d\)
over width \(w\). Should a pointing task be described at a higher level simply
because the size of the target in relation to distance becomes smaller? GOMS
surely requires a clearer statement of what is meant by a level of description than is
presently given.

CCT implicitly assumes levels of knowledge in goal structures and production systems without clarifying what they actually mean, and CLG uses a pragmatic design-oriented justification for its use of levels, again without clarifying what they entail. As has been stated earlier with respect to CCT, all of these approaches need more explicit statements of the factors which might determine the level at which human representational structures and performance might be most accurately simulated or represented.

The vagueness of definitions of levels of analysis or description could be a source of confusion to designers. Methods that rely upon the intuition of the analyst may not be adequate for individuals who have little psychological experience. Moreover if designers are not convinced that they understand and can properly apply a technique, they are unlikely to experiment with it and risk making mistakes in an applied project under commercial pressures.

Whether these techniques can really be generalised to other systems than text editors remains in question. Further research is required on the applicability of HCI DETs to such systems as process control, data processing, CAD, and so on. Green et al (1987) point out that there are a wide variety of applications which probably behave very differently from text editors (and similar systems such as drafting programs, spreadsheets and music editors) which may not be amenable to existing HCI DETs which tend to concentrate on "formal" models of requirements in particular limited domains.

It remains to be proved that the metrics for evaluation supplied by HCI DETs can be made available early enough in design to influence its course. So far all of the evaluations and claims made with respect to these techniques seem to be based upon rather artificial application situations. More analysis of HCI approaches in practice seems to be required before these claims can be accepted.

It is not clear how HCI DETs are to be integrated in applied design given that there are other design activities involved and that various activities are interdependent. Three of the techniques described in Chapter 2 do present explicit views of the system design process into which they assume they will be incorporated. These
views are incomplete as we have seen, leaving much that might be relevant to the
initiative of the analyst/designer. In some instances, these views may even be mis-
taken; for example CLG’s assumption that metrics which are only available late in
the system specification process can be valuable.

In addition a poor view is given of the nature of the supporting activities which
DETs rely on for their basis. Who carries these out, when, and how is not clearly
established.

Currently available HCI DETs suffer from a lack of proven benefits for design
practice. Designers need to know whether these techniques actually work as their
authors say they should in applied design. We need to know whether systems
designers and analysts can actually use them successfully, whether they do actually
improve UI design and can speed up the process or save costs. We also need to
know how they fare in comparison with other techniques, particularly prototyping
and user-evaluation.

What is required is a better picture of what really goes on in UI design practice. It
is all very well to look at HCI theory of design, and Systems Engineering design
approaches in order to judge the applicability of HCI DETs, however such a judge-
ment must be founded upon a clear picture of how real UI design corresponds to the
ideal assumed by HCI Approaches. In the following sections other approaches to
design, both ideal and actual will be compared with HCI DETs design views dis-
cussed above. Knowing more about the reality, we can say more about the prob-
lems which must be overcome by any design technique, be it user-oriented, or oth-
wise. Of course the focus here is the UI, which is typically the poor relation in
structured design methods (Summersgill & Browne 1989). Most well known
SADTs such as SD (Yourdon & Constantine 1978), and JSD (Jackson 1983), give
little consideration to the UI, so they do not feature amongst the user-oriented
design approaches described below.

3.3 Alternative User-Oriented Design Processes

In this section a number of alternative approaches to user-oriented design are
described which explicitly aim to get more Human Factors into actual design prac-
tice. Although by no means an exhaustive survey, the following discussion serves
to illustrate a number of user-oriented approaches which, attempt to address the
design process in particular. In other words, the main distinction drawn in this
thesis between the HCI DETs described above and user-oriented design approaches
is that the former have been developed in response to the need to describe and
evaluate usability of interactive dialogues, whereas the latter have been developed
in response to the need to improve the UI design process itself with respect to users.

Jorgensen (1989) distinguishes two other approaches to design apart from formal
modelling (in the procedural sense used by HCI specialists) which is the basis of the
HCI DETs. He identifies guidelines and prototyping approaches as alternative ways
of addressing the problem of building more usable UIs. These types of approach
are illustrated in the following discussion. They should be contrasted with the
approaches described in Chapter 2 and the earlier part of this chapter, however
space does not permit such a detailed description of these other approaches, the
intention is simply to give a flavour of alternative user-oriented design approaches
for the purpose of comparison with the HCI DETs which concentrate on the use of
specifications of knowledge and procedures for interaction.

The contrast suggests that various groups in the field of HCI appear to be respond-
ing to different pressures. In the case of the HCI DETs the pressure seems to be one
of achieving powerful and predictive techniques which describe users' knowledge
representation or processing properties, and how UI designs impact on these. In the
case of the other user-oriented approaches the pressure seems to be one of impact-
ing the design process directly, since as empirical studies appear to suggest, HCI
seems to have had little effect on current design practice.

3.3.1 Principles and Guidelines

A number of researchers have concentrated on the application of principles or
guidelines which are intended to ensure that UI designs take heed of important user
issues. A general distinction between a principle and a guideline can be made
(Thimbleby 1985) which is that guidelines are applied principles, worked out for a
particular context. Smith (1986) distinguishes between guidelines and standards
which must be complied with. Guidelines offer more flexible and detailed guidance
towards user-oriented design and establishment of agreed design objectives.
Gould & Lewis (1985) propose three principles of practice for user-oriented design which are early focus on users and tasks, empirical measurement and iterative design. These might be considered obvious and rather simplistic but Gould & Lewis provide empirical evidence which suggests that they are not actually obvious to designers.

Thimbleby (1984), Harrison & Thimbleby (1985) outline an approach to design which focuses upon the use of Generative User Engineering Principles (GUEPs) which are formally expressible and are constructive rather than descriptive. GUEPs are distinct from straightforward user-oriented principles such as desk-top modelling and WYSIWYG because they are more directly based upon user characteristics whereas some other principles are more heuristic (based upon ideas that seem to have helped in the past). A GUEP would take the form "you never have to do it all again" (implying that recovery should be possible with some 'undo', or 'again' facility).

Gardner et al (1984) began developing a handbook of Human Factors design guidelines, some of which are now available from the HUSAT Research Centre. They are intended to include detailed guidelines which address specific kinds of UI functionality and documentation for the system.

The documentation guidelines, for example, include identification of areas for documentation, a description of how the documentation design process proceeds, a glossary of terminology for the UI, and detailed guidelines together with explanations and good and bad examples of documentation design solutions. It is interesting to note that the guidelines themselves are written according to their own recommendations.

Glasson (1984) presents a method for deriving a set of guidelines for user participation in the system design cycle. He notes that the variability of different design projects would make it difficult to propose a standard set of guidelines for user participation in all projects. He proposes instead that models for user participation should be derived from the system developers' model of the system development process.

Having defined the tasks required by a systems development methodology the designer is required to build a user model of the design process using a three step
process. The steps involved are; define user actors or user roles required, define the tasks of the actors for each stage of development, and define and develop appropriate procedures, methods, and tools which may be manual or automated, to be used in carrying out the tasks.

3.3.2 Iterative Design and Prototyping

One of the principles of Gould & Lewis (1985) is to incorporate iteration in design as this, they suggest, will significantly improve the design of the UI. Supportive evidence for this hypothesis has been presented by Boehm et al (1984) in an experimental comparison of Prototyping and specifying. However there are a multiplicity of different ways of prototyping systems, and only a small subset of examples can be included here.

Rosson et al (1987) distinguish evolutionary prototyping where the prototype is the evolving system, and simulation or throw-it-away prototyping where the prototype is a disposable simulation used for exploration of possible solutions. Hekmatpour & Ince (1987) also distinguish between two types of non-disposable prototyping evolutionary and incremental. Evolutionary prototyping requires that the whole system is prototyped early on and then increasingly refined into the final product. Incremental prototyping involves building the system section by section.

Christensen & Kreplin (1984) adopt a half-way approach in their attempt to produce an applicable method for software developers in the real world. Their approach, which allows them to specify the dynamics and lay-out of the UI in a largely formal manner, enables them to build prototypes directly from the specifications. The authors admit that their specification language is too low level to be satisfactory and requires an abstraction mechanism to improve specification of the dynamics and lay-out more efficiently. However they claim that their aims of easing communication between designers and users, and improving the user's ability to evaluate designs before they are fixed seem to be satisfied by the approach.

Wasserman (1985), and Wasserman et al (1986) have developed a sophisticated interactive information systems (IIS) development approach called user software engineering (USE). The USE methodology involves simulation prototyping rather than evolutionary prototyping, however tools are provided which significantly speed
up actual system implementation based upon the simulation. This methodology combines executable UI specifications based upon a more extended state transition network (STN) approach than typically used in software engineering with the RAPID/USE prototyping system which contains a transition diagram interpreter which permits the specifications to be run as dynamic UI mock-ups. Users are then able to interact with what essentially amounts to a dynamic specification of the UI (they do not see the state transition diagrams themselves). In this way it is possible for real users to take the place of Kieras & Polson's GOMS model in evaluation of a system, as long as the approach to design follows the USE methodology.

The main problem with the RAPID/USE prototyping toolkit seems to be that the simulations are rather primitive as they do not deal with WIMPS interfaces. This means that the UI simulations may not be able to represent more modern styles of interaction at present.

It is tools such as those in the RAPID/USE toolkit, which appear to be lacking in HCI DETs, which ensure the utility of the approach. Wasserman et al claim that the USE methodology supported by the RAPID prototyping tools is widely used, and provides a highly valuable means for involving users in the UI design process to great effect. He maintains that the methodology would be of little utility without the simulation and implementation tools which considerably reduce the problems of determining usability and the time taken to implement the actual system.

Hekmatpour & Ince (1987) are also advocates of the prototyping approach. They are particularly interested in evolutionary prototyping which generates complete system designs early on in the life-cycle and which they believe to be the most suitable approach for dealing with UI because it aids requirements clarification and permits the design team to deal more easily with inevitable requirements drift. Hekmatpour & Ince present 'Evolutionary Prototyping Language' (EPROL) which is a wide-spectrum executable specification language which can be used for specifications throughout the system life-cycle.

An advantage claimed for EPROL over the USE approach is that it is more adapted to handling WIMPS interfaces. The other main advantage is that it can improve communication between experts with various specialisations. It contains notational devices which enable it to specify the entire system (hence the term wide-spectrum
language) throughout its life-cycle, and can represent the UI from early stages, using executable specifications based upon STNs. These are refined later on to adapt the UI to a WIMPS style of interaction.

Separation of the UI from the application in prototyping as well as executable specifications are ideas which are becoming increasingly attractive as means of speeding up the prototyping process (Adhami et al 1987, Cockton 1987b, Alexander 1987). As support tools such as those described by Wasserman et al become more commonly and affordably available, so the process of prototyping will become more and more efficient and less time consuming. It should soon be possible to test most systems quickly and easily before the functionality is fixed, given that it is possible to get prospective users to try out the prototype. This may not remove the need for analytic HCI DETs but it will surely help to corroborate metrics generated or design solutions adopted through their application.

Jorgensen (1984, 1989) identifies an informal approach used spontaneously by some designers to develop UI prototypes which can be tested by users doing plausible tasks and thinking aloud as they go. The approach was used by designers who each used approximately three subjects for each round of testing of their designs. The thinking-aloud approach appears to be a very useful tool for designers, providing valuable and detailed feedback which enables the designer to alter the system in response.

From the study he conducted Jorgensen supplies a number of recommendations for the approach which include; applying it to mock-ups early in design, as well as to prototypes later; using logging, audio or video recording devices; using about three users (singly or two together for each test round); and conducting interviews after each test round. Although Jorgensen warns that this approach is by no means a substitute for techniques such as task analysis, alpha- and beta-tests etc, it has the value of being easy for systems designers, who are able to prototype their system, to use and the approach is able to be applied at any stage in the development of the system.

User participation in design prototyping is also recommended by Tesler (1983) but he also suggests that users be involved in making design decisions. In addition he emphasizes that the users involved in testing UIs should be 'novice users who are prospective end-users of the system'.
User-oriented design approaches are beginning to gain acceptance amongst certain designers, although these may be the exception, rather than the rule. Recommendations which support the use of guidelines, principles and iterative procedures certainly have their supporters as we have just seen, but the user-oriented views of design discussed above may not reflect design practice in general. Aiming to design new HCI DETs in accordance with user-oriented approaches may still be invalid as a research goal unless we have a clear picture of the reality within which such techniques will be expected to work.

### 3.3.3 Overview

User-oriented design activities are considered as an integrated part of an overall design process. Their input and output are related to the rest of design as a whole. Most concentrate on pragmatic issues such as simplicity, time and cost, how design itself is driven, and they appear to be more general in scope, concentrating less on precision over a smaller range of issues as do the HCI DETs discussed.

Many of the problematic characteristics of HCI DETs are avoided by the avoidance of complexity of user-oriented design approaches, such as that of Jorgensen (1989) are explicitly kept simple to ensure that most designers will be able to apply them. Others, such as that of Wasserman et al (1986) rely heavily on the use of automated tools which considerably lessen the workload on the design team.

Rather than relying on simulations of user representations of interaction rules, such as the grammars supplied by HCI DETs, user-oriented design approaches tend more to rely on direct user feedback, or user involvement in the design process itself (e.g. Gould & Lewis 1985).

### 3.4 Practitioners' Views of UI Design and Evaluation

There is a small but growing body of research into the practice of UI design in applied environments. These studies have a number of aims which include identification of activities involved in real world design practice, identification of inadequacies in UI design, analysis of problems which commonly occur, validation of hypotheses about good design practice, and, rarely, comparison of alternative methods of design. The value of this research is that it can help to direct the efforts
of those building new tools and techniques towards areas where they are most needed. It can also show what kinds of problems these tools and techniques need to address, both in terms of the focus of the analysis they support and the nature of the analysis itself. In the following sub-sections a number of such studies are described, and their implications related to some of the issues raised earlier in this chapter and in the preceding chapter.

3.4.1 Comparative Studies of Design and Evaluation Approaches

Boehm et al (1984) were one of the few groups to conduct a comparative study between alternative design approaches. They compared two approaches; specifying, and prototyping in order to see what were the strengths and weaknesses of each. Although neither approach involved the use of HCI DETs, the specification versus prototyping comparison perhaps suggests that a similar effort might be relevant for HCI techniques. The subjects in the Boehm et al experiment were asked to design and implement an interactive software cost estimation tool. The specifying teams had to produce requirements and design specifications, whereas the prototyping teams built prototype demo’s of the system instead. Although the study suffers from the usual representativeness problems of experimental studies, it does provide some welcome comparative information regarding these competing approaches.

The final product of each of the teams in both groups was evaluated by the experimenters according to a number of criteria such as person-hours spent on the product, maintainability of the code, and so on. Prototyped products were rated to be lower in overall functionality and tolerance of erroneous input, but correspondingly higher in their ease of learning and ease of use. Prototyped products were also rated as markedly more maintainable than specified products. What is important with respect to the discussion here is that prototyped systems had interfaces which were rated as significantly better than the specified systems. This suggests that, discounting inclusion of HCI DETs, prototyping may be superior to specifying as a method of developing UIs. It would appear that such a comparative study might be very useful for determining the value of using an HCI DET as opposed to some other approach such as prototyping. No such study seems to have been carried out for any of the techniques discussed in Chapter 2.

An interesting by product of this study was that smaller teams seemed to be
significantly more productive in terms of delivered source instructions (DSI) per person-hour (even after the results of a particularly productive prototyping team were discounted). This finding implies that activities such as communication and coordination which will be required to an increasing degree as the size of a team increases, take up time in themselves. We should not criticise designers in large teams for being unproductive if the activities associated with ensuring shared goals and common concepts are necessarily taking up time (for instance communicating results of a task analysis to a systems analyst). However there may be something to be said for avoiding unnecessarily large design teams, and poor communication channels.

Maclean et al. (1985) compared the use of two different empirical metrics of usability; the time taken to complete tasks (as is used by GOMS) with the method users choose to accomplish tasks (where the assumption is unlike that of the GOMS approach; i.e. that users do not always choose the optimal method). Although this is not a comparison of full approaches it addresses the application of metrics which many performance modelling approaches claim may be used as indicators of usability. It was found that the subject system users often used slower, less efficient methods although they believed they were actually using the fastest methods.

Although the study was aimed primarily at analysing why users choose a given method when another might be more efficient, it highlights a very important issue for HCI DETs, namely that assumptions about users' behaviour made by models of their task execution may be ill founded and unrepresentative. The selection rules, applied by users when there is more than one way of accomplishing a task, may be none ideal. This finding is most relevant to performance models such as GOMS and CCT which, although they incorporate selection rules, are not sensitive to qualitative features of UIs which may cause users to choose particular non-optimal methods for accomplishing tasks. The findings are also relevant to competence models such as Reisner's Action Language and TAG since they assume competence in their models, rather than identifying realistic users' task methods which may be less than ideal. In fact Maclean et al produced a useful technique for helping an analyst to predict the methods users might select on the basis of features of the task environment. Such a technique might be a valuable component of future HCI approaches.
3.4.2 Studies of HCI Principles Applied in Practice

Gould & Lewis (1985) state that in the 1970's they began recommending what they refer to as three principles relating to methods for UI design (as opposed to the ideal UI-property oriented principles described in Chapter 1). These design principles were:

- Early Focus on Users and Tasks
- Empirical Measurement
- Iterative Design

When presented with these principles, people seemed to think that they were rather obvious, so the authors carried out a survey in which they asked designers to write down a sequence of about five steps one should go through when developing and evaluating a new computer system for end users.

There were 447 participating designers in the study who were attending a human factors talk, so one would have expected them to be at least as aware of user requirements as average. However only 16% mentioned all three principles in their steps, 24% mentioned two, 35% mentioned one, and 26% none. Early focus on users was mentioned by 62 participants out of the total, empirical measurement by 40 and iterative design by only 20 of the participants.

Some designers were given credit for including principles when in fact they did not really have what the authors considered to be a good idea of what the activities specified by the principles really involved. Gould and Lewis argue that designers appear to have a number of misconceptions about good design practice, the value of user involvement as opposed to other approaches, the amount of time taken and cost required to ensure usability, and so on.

In a later study Gould et al (1987) report the attempt to apply the three principles of design in practice on the 1984 Olympic Messaging System (OMS). They found, as they expected, that it was never easy to get things right first time. Extensive and careful evaluation was necessary with early demonstrations, and simulations having to be built. A number of compromises, such as simplicity at the expense of functionality, had to be made. However, at the end of the project, the success of the product convinced the team that the principles did work; they found that they had a robust, reliable and usable system.
Based upon the project, a fourth principle was instantiated: Integrated Usability Design which demands that all usability factors should evolve together, and that all aspects of usability should be under one control, rather than groups trying to deal with them autonomously, and possibly in a different fashion. Usability factors were seen to interact in the OMS design project, so it was impossible to develop and evaluate them in isolation. The need for integration imposes the requirement that one person or group takes overall responsibility because it is then easier to ensure that what needs to be done does indeed get done properly.

Another study covering human factors principles in general was carried out by Grimes et al (1986) to discover whether these principles are well or poorly integrated into design. Systems designers completed a survey questionnaire issued by the researchers at a SIGCHI conference. The questionnaire required the respondent to describe their position, background, and experience with system design, and how human factors issues were integrated into the UI design process.

The authors found that the companies of these designers varied widely in the degree to which they were rated as integrating human factors with systems design. However the results suggested that human factors is generally poorly integrated in the majority of companies at present. As might be expected the researchers found that only a few of the designers had a psychology background (which many of the HCI DETs discussed in Chapter 2 seem to expect), with the majority being pure computer scientists.

The implications from Gould & Lewis's work and that of Grimes et al, is that designers, possibly due to their lack of experience with psychological and HCI perspectives on design, generally have a poor understanding of HCI issues and that they tend to underestimate the importance of human factors in their practice. This bodes ill for HCI DETs which could be perceived as difficult and time consuming by designers. If the potential users of HCI DETs do not value the targets of these techniques (i.e. detailed analysis of the UI with respect to various user characteristics which could affect interactive performance) they may not be prepared to invest the time and effort required to carry them out.

Mosier & Smith (1986) carried out a survey of the users of a "comprehensive compilation of design guidelines"; the Smith and Aucella guidelines (Smith & Aucella
1983) which are provided in a report containing some 580 guidelines. They found that managers and software designers were far less likely to read guidelines thoroughly than HCI specialists, and software designers seemed to find the guidelines less helpful than HCI specialists and others involved in some aspect of design (such as teachers of methods, and systems analysts). It may be that a lack of incentive and psychological experience prevents designers from understanding the actual meaning of guidelines, or that designers do not appreciate the importance of guidelines and their relevance to the success of the end-product. The lack of use may also have been because of the difficulty of application of guidelines to software design practice.

The main problems with guidelines were that they were often found to be not relevant to the system in question, inapplicable for practical reasons, or too general to be useful. It would appear that guidelines must be carefully designed to strike a balance between generality and precision. These problems almost undoubtedly will be relevant to HCI DETs if and when they are applied by designers. Since most DETs have not been tested in practice as have guidelines, it would be unwise to assume that they do not share many of the same problems.

3.4.3 Studies of UI Design Practice In General

Dagwell & Weber (1983) carried out an international investigation of systems designers' models (assumptions) of their prospective using a questionnaire survey. They viewed the responses they received as being indicative of the fact that designers' models of prospective users of their systems are possibly inadequate; the designers in the study did not consider how great an impact they have in terms of changing people's work, both on the individual and on the organisational level, and some of their methodologies were outdated and naive. They considered that better, context sensitive user models would be of great value in design, perhaps being conveyed through education and experience. Studies such as this indicate that designers do need supporting tools to help them improve their sensitivity to their prospective users needs, beyond simplifying their systems to the extent that users jobs are completely deskilled.

Hammond et al (1983) note that much needed HCI literature runs the risk of being inapplicable for systems design, with overgeneralisation of recommendations based
upon artificial experimentation. The authors identified a need to find out more about the decision making processes which influence the design of a UI. For this reason they interviewed five designers in a very large systems design company.

From the interviews Hammond et al were able to outline a framework, shown in figure 3.3, which the interviews suggested was typical of the design process. They also used the designers' comments to illustrate ways in which the designers considered users and their tasks.

Figure 3.3

Although aware that user requirements had to be given consideration, the designers
tended to base their task designs around the architectures of their designs and the logical substance of the tasks involved rather than on users’ natural methods or the most efficient methods for the user. Compatibility as a system feature was also problematic for designers. Compatibility with previous systems the user might have experience with could have been an excuse for maintaining outmoded UI components. Sometimes this concept seemed to be used as an excuse for imposing unusable features of the UI on the design.

Typically designers theories about users were not sensitive to task and user variables which could have strongly influenced performance. They were based upon general experience or "common sense" rather than any formalised body of knowledge. Designers views of human factors were based upon very little knowledge and no small amount of disrespect which could have been due to poor company organisation of its human factors input to design.

Hammond et al state that expressions of dissatisfaction with human factors and HCI practitioners were probably due to "organisational and resource constraints". As stated earlier HCI DETs require considerable expertise and time to apply. It would appear unlikely that such techniques would be applied in the circumstances of the company studied by these researchers.

Smith & Mosier (1984) reported a survey which revealed that Human Factors engineers in mainly industrial settings estimated that between only a fifth and a third of system development projects adequately considered USI requirements at various stages of the development process. Together with this finding a significant (approximately 7%) of the UIs in question were not given any consideration at all in the system specifications. This finding supports the relevance of the warning given at the beginning of this chapter that in some circumstances a designer may not be obliged or motivated to take any steps to ensure usability of the system.

Over half of the survey respondents to the appropriate questions considered that UI documentation was inadequate, and most of the respondents did not use guidelines. The authors suggest that the blame for poor UI design approaches may lie at the door of HCI specialists who lack the knowledge and tools, and consequently the influence, to deal effectively with design. This criticism must also apply to HCI DETs which must embody knowledge about design practice and tools to support
time consuming activities required.

More recently Rosson et al (1987) conducted an interview study based upon the idea that, if specialists are ever to provide useful UI design tools, they have to look at design practice. Twenty-three systems designers, with a very wide variety of backgrounds (but mainly from one organisation) were interviewed and derived a great deal of qualitative data from this study. The projects involved a number of types of application including office support systems, tracking systems, information or function access, and software development support. The products would run on mainframes or in intelligent workstation environments.

They noticed that the projects they studied were split almost half and half between phased development in which there was a design phase followed by an implementation phase (10 projects) and incremental development with design and implementation proceeding in parallel in an iterative fashion (12 projects) more commercial projects used the more tightly controlled phased development design approach, whereas research projects tended to use incremental prototyping. Each approach had its drawbacks, notably phased development was seen to limit iteration to the early stages of design, whereas incremental prototyping involved less certainty about the form of the potential end product.

User testing was mostly informal, ranging from active user involvement, to belated user testing, however no special effort seemed to be made to ensure that representative users and tasks were selected for evaluations. Design was assisted by demonstrations and prototype testing but lack of user feedback early in design was a common complaint of designers. This was blamed on lack of adequate information about prospective users' needs, lack of prototyping tools, lack of resources, product confidentiality, and test administration group problems. Prototypers did not always exploit their prototype claiming that user testing would force them to waste time improving robustness of the system too early on. This indicates that simulation tools could be helpful. Separation of the UI design from the application was adopted by some teams a few working from the outside in, starting with the UI, as Wasserman recommends.

Surprisingly designers using incremental, prototyping approaches were not more likely to offer an early interactive prototype for user than were designers using
testing phased development approaches. This seems to be attributable to the lack of robustness of early prototypes.

Amongst the recommendations derived from their study Rosson et al state that UI designers require a rich, modular prototyping environment (which would be highly modifiable) for evolving a final product. This will enable them to deal with requirements drift and unforeseen changes which are often unavoidable in design.

Hannigan & Herring (1987) report on a study of the experience of designers from five major manufacturers. They compared design practice with generic design cycle models both from research and from the companies themselves and found that extreme deviation from these models was the rule rather than the exception.

This extreme variation is viewed as being characterised by multiple views of the design by different stakeholders in the process. The authors of the report state that it is unlikely that this state of affairs will change and that HCI input will have to adapt to the design process rather than vice versa. The current state of affairs seems to be that, although UI design advice is available, it is not used because its form is not directly applicable. The same may be likely for HCI tools and the authors support a view expressed earlier in this chapter, stating that "if the cost in terms of access, time to use, up front learning effort etc, outstrips any perceived benefit they will not get used".

The study revealed that task analysis methodologies were never stated as a major designer requirement (unlike advice). On the other hand more detailed analysis showed that designers did acknowledge that they needed realistic task scenarios or simple methodologies. Hannigan & Herring warn that such methodologies will be subject to the same constraints as other human factors inputs to the design cycle noting external factors such as markets and internal factors to do with organisational culture.

User representation in the design process itself was widely accepted but not well understood by designers. Users requirements in specifications were not given formal consideration as were other design aspects. Also the focus of attention was directed towards evaluation, instead of design with casual ad hoc approaches being adopted.
The evidence from the studies in the preceding discussion suggests that, in spite of the amount of knowledge which already exists in the field of HCI, designers do not generally approach UI design in an ideal manner. Guidelines are sometimes used, but in some cases even these tailor made, simple design aids are ignored and no consideration is given to user requirements in design.

As has been noted, UIs can evolve from systems without having been designed, that is to say without restriction on what might be considered acceptable. However the evidence suggests that this is rare. On the other hand it is far more common to find that designs are not influenced by the kinds of restrictions which HCI research and techniques could provide. When users are considered, it appears that designers adopt a casual, ad hoc approach placing more emphasis on evaluation than on design for users.

It is not surprising that Gould and Lewis (1985) found that designers thought they were applying user-oriented design principles when, in fact, they were not. It appears that there is a common attitude that as long as users have been considered then their needs have been addressed. This belief must surely be wrong because there can be no sensible basis for assuming that designers’ intuitions about users are always correct. What is clear above all from the literature is that there is certainly scope for improvement.

3.4.4 Overview

The evidence supplied by existing design practice studies gives rise to a number of interesting implications regarding actualities of design practice. The first is that prototyping may represent a good way of improving usability of a UI over what can be achieved using design specification techniques (Boehm et al 1984), if only problems relating to lack of adequate prototyping tools which are both robust and modifiable can be solved. The value of prototypes seems to be that users who know most about their own requirements and limitations can see actually what the system is like, make suggestions, and provide valuable feedback into the design process (Gould & Lewis 1985, Gould et al 1987).

Evidence from Maclean et al suggests that attempts to model interaction based upon idealised user behaviour of the sort assumed by GOMS models may be misguided.
Their findings suggest that users often believe that they are using the fastest most efficient methods for achieving their tasks, when, in fact, they are not. If this is the case, then only intensive procedural training, such as might occur in the armed forces, is likely to achieve idealised efficient performance.

The findings from Gould and Lewis (1985) and Mosier & Smith (1986) suggest that attitudes of designers may be problematic for the application of HCI DETs. Gould & Lewis show that designers assume that they are adopting user-centred design approaches, when in fact they are not. Mosier & Smith reveal that software designers and managers are far less likely to read design guidelines than HCI specialists. Smith & Mosier (1984) show that user considerations are often neglected or ignored altogether.

Grimes et al (1986) demonstrate that most system designers lack experience with HCI which may make it all the more difficult for them to assimilate and apply HCI guidelines, principles and techniques. Mosier & Smith's study and that of Hammond et al (1983) point some of the blame for poor integration of HCI into design at irrelevant unpractical or overgeneral guidelines and HCI specialists who are poorly educated about systems design.

This is not to say that HCI tools cannot be useful and will not be useful for more commercial projects if appropriately designed. Evidence from Dagwell & Weber (1983) and Hannigan & Herring (1987) suggests that they can be. However, as Ratcliffe (1987) points out "All projects are subject to constraints of various kinds - temporal, financial, technological, and legal amongst others." Such constraints will have to be overcome, and attitudes to acceptability of "common sense" solutions or excuses for implementing outdated design solutions, or basing interaction around system hardware and software, rather than vice versa must be changed.

Rosson et al (1987) point out that commercial design is also constrained by development approaches adopted by organisations. All too often, informal and belated user testing results in inadequate evaluations. Designers blame inadequate resources and lack of prototyping tools. However even when such tools exist it appears that they are underexploited.

In general designers appear to have positive attitudes to HCI in principle but not so
much in practice. They also suffer from lack of experience and understanding of HCI's value and methods.

The additional principle of Gould et al (1987); *Integrated Usability Design* is based upon HCI practitioners practical experience on a real design project. It is likely that HCI specialists with experience of systems design themselves are likely to produce very relevant advice for others. It makes sense to assume that responsibility for a difficult system feature to pin down will be better addressed if there is one control; in essence, the buck has to stop somewhere. As has been pointed out in the preceding chapter, the fact that different aspects of usability interact with, and may compromise one another means that integrated evolution must be a better way of identifying possible trade-offs as soon as they emerge. Such an approach will help to ensure that the necessary modifications will be identified earlier, and be easier to make for that very reason.

3.5 Discrepancies Between Theoretical Design Views and Design Views Based on Studies of Practice

There are a number of discrepancies which can be identified between theoretical HCI DET views and views of UI design which come from studies of actual practice. Perhaps the most striking is that empirical evidence suggests that, although they are generally positive about user centred design (Gould & Lewis 1985), systems designers do not typically address user issues as rigorously as required by HCI DETs. A great deal of analysis is necessary to support many DETs, for example TAG requires that simple tasks be identified, listed and analysed to generate rule schemas. This may be simple for some applications but for others this requirement may be extremely time consuming, and at present the evidence does not support the view that designers are generally prepared to carry out such detailed and scientific usability analyses (Rosson et al 1987). In fact it would appear that even tailor made, relatively simple guidelines are not thoroughly read by software designers (Mosier & Smith 1986).

HCI DETs do not come with a clear explanation and justification of the design process into which they are to be integrated. Evidence suggests that the design process is always varied. Some techniques do not address this fact at all, others only claim to be research tools. Organisational and resource constraints or more
detailed aspects of the assumed design process are left unaccounted for. For example CCT claims to be an early evaluation technique, but it implicitly assumes a great deal of resources are available to the design team for building a detailed UI simulation before going on to build the actual system. It is not clear that the metrics supplied will always be available before the application software is constructed and that many design projects will comply with assumptions made by CCT.

Evidence of the utility of HCI DETs over competing UI design and evaluation approaches is lacking. Sharratt’s (1987) study suggests that HCI DETs may not be cost effective. Evidence from studies of design practice indicates that designers attitudes to HCI may not be positive enough for them to invest time and effort in discovering and learning how to apply relevant approaches for their projects.

Validations of the accuracy of HCI DETs predictions are carried out by HCI experts. The assumption of expertise by these techniques may be unjustified for the majority of design projects. Accuracy of a grammatical model or performance model of interaction may rely to a great extent on the experience the analyst who applies it, rather than on explicit application methods associated with the technique itself. The evidence from real world systems designers is that they are not well informed about HCI and psychology.

Few HCI DETs address the value and implications of prototyping in detail. Iteration is acknowledged by some approaches, but how this is dealt with is not clear. Moran, for example, admits that more detail needs to be supplied by CLG as to how iteration is dealt with by the technique. On the other hand, in practice, prototyping appears to be a very common practice (Rosson et al 1987) and therefore must be seriously considered by applicable HCI techniques.

Competence models assume ideal users which may be unrealistic in most circumstances. Maclean et al (1985) demonstrate that most users cannot judge the efficiency of their methods. Payne & Green (1986) point out that acceptance models which would include non-ideal methods for accomplishing tasks would have to include so many alternative methods that they would be too large to build in a sensible amount of time.
3.6 Summary

The most important general point to make here is that it may well be that it is not entirely the fault of systems designers that they are not applying HCI knowledge. Some systems designers seem to be supportive of the idea that user-oriented design is a good thing (e.g. Hammond et al 1983). However some researchers are beginning to suggest that HCI techniques and guidelines are themselves inadequate as design support tools (Barnard 1986, Smith & Mosier 1983, Mosier & Smith 1986, Green et al 1987).

The most striking fact is that there seems to be almost no literature relating to the use of HCI DETs of the type discussed in Chapter 2 in commercial practice, in the hands of individuals other than their developers. Sharratt (1987) represents a notable exception in this general state of affairs. Any criticisms relating to the applicability or otherwise of these techniques may therefore rely largely upon inference. Some suggestive evidence is available from empirical studies which address designers attitudes towards guidelines. As we have seen guidelines, which are what Thimbleby (1985) refers to as principles worked out in practice, are themselves often found to be inappropriate or too general to be of use. There seems to be some suggestive evidence (Mosier & Smith 1983) that guidelines are more useful to those with a psychological background. HCI DETs are considerably more complex and rely even more on psychological expertise. It seems highly likely that they will be subject to the same criticisms and more.

It seems that a major rethink is required from HCI theoreticians who are attempting to generate techniques to be used by others, particularly psychology naive systems designers. A better understanding of designers' requirements is needed by HCI researchers, just as a better understanding of users' requirements is needed by designers. HCI is, in a manner of speaking, guilty of the same crime that it is attempting to solve; that is not considering its users.

In order to understand designers requirements, given that the process of design is so enormously variable (Gould & Lewis 1985, Hannigan & Herring 1987) we must look at design practice in the real world. A better picture is necessary of the constraints which designers have to operate under and of the kinds of activities involved and problems which emerge. Having a better view of real design will help
HCI theoreticians to identify the kinds of problems their design and evaluative techniques must address.

Hannigan & Herring (1987) reveal that simple models of UI design will not be sufficient for the purpose of integrating guidelines and HCI DETs. There is too much variation to present any coherent picture of standard design practice. Perhaps what we should be looking at are features of UI design without trying to impose a rigid structure over them. Identifying features of design such as activities involved, necessary sequences, constraints, problems, and so forth, is better than having no view at all, or a view which represents only a tiny minority of situations.

3.6.1 Questions Raised by Varied Design Views

The literature available on design practice, does not generally address specific discrepancies between HCI DETs requirements of design evinced by the assumptions they make and the design views they have. It appears that much of the literature focuses more on user-oriented design practice in general. This means that a number of issues raised in the preceding discussion still need to be addressed.

1. Are designers prepared to devote the time, effort and resources necessary to apply HCI DETs. Do they, or can they, collect the kind of data required to support HCI DETs (such as task analyses, or user characterisations).

2. Are designers well enough educated regarding psychology and HCI to appreciate the importance and relevance of various techniques available. Are they able to select, adapt and apply these techniques themselves.

3. Are abstract specifications themselves sufficiently comprehensive to provide a good basis for evaluation since they do not convey the look and feel of the system to designers or users.

4. Can levels of analysis required by an HCI analysis be identified a priori to analysis itself.

5. Are there HCI DETs suitable for application to most design projects, in terms of user characteristics, UI type, application, and target tasks. Are the metrics
supplied by HCI DETs likely to be widely applicable for UI evaluation.

6. Can HCI DETs in their present form be integrated into current software design practice, and if not how can they be adapted for integration.

There is little, if any, evidence to support the argument that HCI DETs are or can be used in most design situations. The majority of research directed at validating claims that these techniques are useful for design is centred around application of techniques by experts to selected existing designs in artificial circumstances.

Discrepancies between theoretical views of design and those views based upon design practice in the real world support a number of tentative hypotheses with regard to integration of user-oriented design approaches into current design practice.

In the following three chapters these questions will be addressed. Three studies of applied UI design practice will be discussed which attempt to provide information about the features of commercial design practice which might impinge upon HCI DETs. The first is a questionnaire based study which seeks to provide a picture of the variety amongst design team structures in applied and commercial practice. It also attempts to identify activities undertaken by design teams in order to ensure usability of their UIs. A number of hypotheses about the nature of design and its results are examined.

The following two interview based studies take broader approaches, the first focusing in more detail on how designers carry out UI design, are constrained by their environment and on what problems they experience. The second looks at HCI practitioners, how they influence design, are constrained, and on what problems they have to deal with. It is hoped that by comparing systems designers with HCI specialists we may gain valuable insight into applied UI design practice.

Knowledge of how systems designers’ lack of HCI expertise compares with the expertise of specialists in the field, and what difference this makes in terms of success, may be valuable input to future HCI DETs which must supply the missing expertise, together with methods and tools that are well designed to address problems which will emerge within the constraints which exist in commercial design practice. Only by doing this can these techniques be sure of being applicable and
useful.
Chapter 4

A Features Analysis of Design Projects:
Designing Systems for Users.

4.1 Introduction and Rationale

The reviews of HCI DETs in Chapter 2 and of UI design views in Chapter 3 suggest that there are a number of questions and discrepancies between theoretical and empirical, user-oriented design views which need to be addressed by applicable user-oriented and UI design and evaluative techniques. This chapter describes a questionnaire based study which focuses on features of design practice in the real world, as opposed to the HCI specialist's laboratory. The study looks at activities, design constraints and information sources in applied and commercial systems design projects.

Hypotheses Based Upon Discussions So Far

A number of hypotheses are proposed below which relate to the nature of the HCI DETs and the extent to which they may or may not be applicable. Personal experience of the author, informal discussions with systems designers at Queen Mary College where the research took place, the apparent discrepancies between HCI DETs design views or assumptions and actual design practice, and other questions raised by assumptions about design practice embodied in HCI techniques described in Chapter 3 all contributed to these hypotheses. The features analysis of real world user-oriented design practice reported in this chapter attempts to address them.

The hypotheses are assertions questioning the validity of various claims made by the developers of HCI DETs which do not seem to have been empirically validated.

1. The design process is very variable, in terms of activities and organisation.

2. Applications are diverse and text editors only make up a small minority of these
(this hypothesis queries the generalizability of HCI DETs to a wide range of UIs).

3. Prospective user populations are highly variable, and the existence of ideal or even expert users cannot be relied upon.

4. Time and project resources are frequently insufficient for a satisfactory amount of work to be done on the system design itself which affects the UI design also.

5. Design teams frequently lack HCI or psychology expertise as would be required for instance to identify appropriate levels of analysis in a multi level HCI DET.

6. Abstract design specifications of any type (including systems analysis and design methods as well as HCI methods) are not commonly used.

7. Prototyping is a very common feature of design projects.

8. Designers are not rigorous in ensuring user-oriented design and tend to favour casual and late evaluation, rather than user-driven design and early evaluation.

9. There are many constraints which could pressurise design teams towards certain methods and away from novel unproven methods.

As well as determining whether these hypotheses are valid, this study seeks to provide some tentative explanations as to why they are, or are not, supported. The findings in relation to these hypotheses represent a number of features of applied and commercial design practice which are relevant to the application of HCI DETs and development of future techniques. These features are discussed in the discussion section (see 4.5) of this Chapter.

4.2 Questionnaire Structure and Methodology

A questionnaire study was undertaken to identify the features of applied and particularly commercial user-oriented and UI design practice. Questionnaires were the chosen data collection method in order to capture as wide a range of individual projects as possible. The questionnaires stipulated that the respondent refer only to a single design project in order that a representative picture of the activities undertak-
en be given. The assumption was that each individual project involves only a subset of the activities which designers undertake in general (it so happens that this assumption was supported by one respondent who answered part of the questionnaire referring to practice in general and included far more activities than most other respondents).

Respondents were encouraged to refer to commercial design projects rather than research projects since research projects were expected to be less representative of practice in general, even though they might be more interesting and productive.

The design of the questionnaire was aimed at encouraging respondents to provide information which might be of an unpredictable and varied nature, since existing evidence suggests that design practice is extremely diverse in nature (Rosson et al 1987). For this reason the questionnaire was not multiple choice in style, and, starting without preconceptions, did not encourage designers to classify their responses according to a predetermined scheme.

Since the questionnaire was clearly asking for information which might suggest the rigour with which designers had pursued usability of their system, the questionnaire avoided asking leading questions such as "Did you use X technique to improve your design". Where such information was sought the question would be more indirect such as "What techniques of Y type did you use; e.g. Acronym A, Acronym B, Acronym C." The reason for avoiding being too specific was that the questionnaire sought to avoid cueing the designer as to what the analyst might consider good design. If the designer did not know what "Y" meant, or had a different definition to that presupposed, then this was also valuable information.

### 4.2.1 Structure of the Questionnaire

The questionnaire included a brief introduction which explained what aspect of design, namely the UI, it focused on (see Appendix 1). It stated clearly (twice) that answers should be based upon a single project only, otherwise the information given would be misleading. The rest of the questionnaire consisted of two parts. The first was a purely descriptive section in which respondents were asked to describe various aspects of their design, organisations involved and activities undertaken (see Appendix 1). The second was aimed at identifying constraints (such as those
identified by Hammond et al 1983) which impinged upon UI design activity and the available information sources which were the basis of UI design decisions. The last part of the questionnaire invited designers to give a subjective assessment of the good and bad features of the final product of the design.

Respondents were asked to state how familiar they were with HCI as a field and to state whether they thought it could be useful to UI design. The full meaning of the acronym was deliberately not given so that it would not cue designers as to what constituted a good or bad response. If designers did not understand the meaning of the acronym but were rigorous as to usability, the questionnaire would pick that up. The point of this was to inform as to the penetration of HCI into actual design practice as a discipline rather than a way of casually assessing a system.

Designers were invited at all stages to give any additional information which they considered might be relevant and to add comments or reasons for their answers.

In more detail, part I asked the respondent to briefly describe roughly the following general aspects of the design (see Appendix 1 for more detail on the precise questions):

- Product Functions and the Type of UI involved
- The Prospective Users
- Organisations involved
- Design Team Size, Roles and Structure
- Availability of Information from Specifications, Task Descriptions, Users
- Design Specifications and Methodologies Used
- Generation and Testing
- Problems and Modifications
- Finalisation and Satisfactoriness of the Final Product
- Familiarity and Attitude Towards HCI as a Discipline
- Additional Information

Although logically Familiarity and Attitude Towards HCI as a Discipline might be the first question, this information was asked for at the end of the first section in order to avoid discouraging respondents who were not familiar with this discipline from answering the questionnaire.
Part II was more structured; it involved 5 sub-sections. Section 1 asked the designer to read a list of possible constraints which might affect design activities and to rank any which they themselves experienced on their project in descending order of importance. They were then asked to add any others and indicate importance relative to constraints from the list.

Section 2 used the same structure as section 1 in that respondents were asked to read a list of possible information sources and rank those which they used in descending order of importance, adding any others not included in the list. Section 3 requested that respondents describe how they went about exploiting each of the information sources to which they had access.

Sections 4 and 5 were less structured in that they asked respondents to give a subjective evaluation of the good and bad features of their UI. This section was not intended to suggest whether the final product really was either good or bad, it was intended at getting a realistic picture of what designers tend to think of as good and bad features of their own UI designs as opposed to features of ideal UIs which would be unconstrained by reality.

4.2.2 Distribution of the Questionnaire

The questionnaire was informally piloted on an HCI specialist and an HCI naive Computer Programmer. These individuals represented almost opposite extremes in the profession of UI design in terms of familiarity with HCI. It was then sent out, in two releases, to approximately 150 addresses in total which were selected largely at random from Queen Mary College's Computer Science Department address database which includes names of contacts of anybody in the Computer Science Department. University addresses were avoided in the first release but not in the second, when most large companies from the QMC database had been contacted.

Although the questionnaire was long and rather complex (it took approximately one hour to complete), 27 largely complete copies were returned (roughly 18% of those sent out). One of these had to be discarded since it was completely illegible, another two clearly came from two different collaborating organisations on the same very large government subsidised project; these were treated as one response for project data, and as two responses for data on constraints experienced and information
sources exploited. Consequently 25 projects contributed to this survey.

4.2.3 Analysis

The responses yielded largely qualitative data, although the rankings of design constraints and information sources provide some statistical evidence for the prevalence of certain constraining circumstances and activities in design. The design of the questionnaires and their analysis was directed at features of design practice rather than a more integrated analysis of design processes and their organisation (a more integrated view of real world design processes is provided by the second interview-based study). However, any information provided by designers as to organisation of the design process has been included in the analysis.

The responses, as has been indicated, were not multiple choice because this was considered to be likely to encourage designers to apply artificial classifications which they might not normally use, and which might be misrepresentative. Consequently responses which involved descriptions had to be assessed and classified when returned. This arrangement probably resulted in a greater number of possible classifications than a multiple choice responses one would have. Equivalent responses were compared and in most cases classified for all 25 projects involved in the study.

The ranked constraints and information sources, where the two responses from the same multi-site project were treated as separate data points, were simply analysed according to two alternative ranking schemes. The first scheme used a simple incidence ranking which assessed the occurrence of constraints and information sources exploited. The second scheme incorporated the ranking of each item as well as occurrence, giving an assessment of the general perceived importance of constraints and information sources to the design process.

In addition to the classification of responses, and ranking of design constraints and exploited information sources, other qualitative information was collected, involving notable occurrences described, and a large number of informative quotes.
4.3 Basic Findings from Descriptive Responses to Part I

4.3.1 Grouping 25 Projects According to Host Organisation

Initially the responses were classified into host organisation groups. Twenty-five different projects were involved in the survey. The host organisation type was viewed as likely to influence the kind of project undertaken and the way in which it is tackled. These three groups are shown in table 4.1 which represents the total number of design projects under each classification, together with the number of projects out of each total involving an HCI specialist. The two respondents from the same multi-site project were in group B and are represented as one response in the table. The classification provides a general view of the nature of the host organisations of the projects involved in the study.

Although the questionnaires were distributed as widely as possible and targeted towards commercial organisations of the type in Groups A and C, recipients tended to redistribute or pass on copies so that a significant number of responses seem to have come from R & D groups (usually associated with universities), universities themselves, or from HCI consultancies (who were not regarded as a target for the questionnaire since they could bias the view of design strongly towards user-oriented practice). It is interesting to note that the prevalence of project team members with knowledge of HCI appears to be extremely high (100%) in Group B (both of the collaborating respondents reported HCI specialists in their project), even though only one or two of the teams belonged in what could be classified as HCI consultancy organisations.

It is not possible to judge how representative the responses are of design practice, but it would seem sensible to assume that response, being voluntary, were self-selected and may reflect a more positive attitude towards HCI and research oriented activities than is the norm.

Another reason for making the above classification is that it may be interesting to see whether any interesting differences emerge between the groups in terms of the features of their design projects. If no differences are apparent, then it may be possible to assume that the type of organisation is of less importance to design practice than within organisation structures. If differences do emerge, then it may be safe to
assume that generalisations from one group to another should be avoided by future research.

Table 4.1
Classification of Design Teams' Organisations

<table>
<thead>
<tr>
<th>Type</th>
<th>Group A</th>
<th>Group B</th>
<th>Group C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Companies &amp; Consultants</td>
<td>12</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>R &amp; D Groups, Universities &amp; HCI Consultancies</td>
<td>8</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Departments of Non Computer Companies</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total number of projects in Group</td>
<td>12</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>Number of projects involving people with knowledge of HCI</td>
<td>5</td>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>

Individual questionnaire responses are referred to as numbered group members; thus 3A would be a member of group A. The numbering serves no other purpose than to distinguish responses from one another.

4.3.2 Project Size

The size of the design teams and the number of person-months devoted to the project was also of interest. As was hypothesised, the amount of time and people working on the project should be related to the ability to fund better user-oriented design approaches. Also larger projects should be more likely to include HCI specialists, as opposed to designers with some lay experience of UI design. The size of the design team and project length were given by 24 of the 25 projects. One of the respondents in group B did not include this data. In some cases respondents indicated the number of person-months spent on the design. Where person months were
not given they were calculated from team size and project duration.

The 24 team sizes are given in table 4.2, and the 24 project lengths in table 4.3.

**Table 4.2**
The Sample of Twenty-Four Design Team Sizes

<table>
<thead>
<tr>
<th>People in Team</th>
<th>Number of Teams</th>
<th>Involvement of HCI Specialist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>in 1 Team</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>in 1 Team</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>in 1 Team</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>in 1 Team</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>in 1 Team</td>
</tr>
<tr>
<td>c30</td>
<td>2</td>
<td>in 2 Teams</td>
</tr>
<tr>
<td>c50</td>
<td>1</td>
<td>in 1 Team</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>8.75</td>
</tr>
<tr>
<td></td>
<td>Median</td>
<td>5</td>
</tr>
</tbody>
</table>

The large differences between the average (8.75) and the median (5) values for
Table 4.3
Twenty-Four Project Lengths in Person Months

<table>
<thead>
<tr>
<th>Person Years</th>
<th>Project Lengths in Person Months</th>
<th>No. of Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up to 1 Year</td>
<td>0.67 2.0 2.5 5.0 8.0 12.0</td>
<td>6</td>
</tr>
<tr>
<td>Up to 2 Years</td>
<td>14.0 14.0 15.0 24.0 24.0</td>
<td>5</td>
</tr>
<tr>
<td>Up to 10 Years</td>
<td>30.0 36.0 48.0 60.0 96.0 120.0</td>
<td>6</td>
</tr>
<tr>
<td>Up to 100 Years</td>
<td>144.0 144.0 180.0 300.0 720.0</td>
<td>5</td>
</tr>
<tr>
<td>Over 100 Years</td>
<td>1440.0 1800.0</td>
<td>2</td>
</tr>
</tbody>
</table>

design team sizes in table 4.2, and particularly project lengths in table 4.3 are due to a few extremely large projects which skew the averages to give values which are probably larger than the great majority of projects values.

The shortest project is 0.67 person-months long; approximately 20 days. The longest project is 1800 person-months long; 150 person years. The average length of projects is 218 person-months long; just over 18 person years. The median is 33 person months or 2.75 person years. No notable relation was identified between organisational group and design team size and project length, with groups. A statistical validation of the data was not undertaken because the few very large projects tended to skew data so greatly that larger samples including a more reliable proportion of large projects would have been needed to conduct a representative statistical analysis. The data suggest that a design team of five members working on a project lasting 6.6 months (33 person-months) is about the most typical design situation.

Eight out of the 25 design projects surveyed were spontaneously assessed by the
respondents as being too short on time (one of the 25 had not stated how long the project was). The actual number of person months involved was not related to this assessment with, amongst others, the third longest and the third shortest being subject to this view.

4.3.3 Familiarity with HCI

Respondents were asked to state the roles of individuals in the design team. A total of 9 projects were found to include HCI specialists (one of the nine was the one which provided no team size and project length data, hence its non-appearance in table 2.4), and it was these individuals who usually answered the questionnaire; presumably the questionnaires were often passed on until HCI specialists received them. Some other projects included team members with UI design experience and lay knowledge of HCI (see table 4.1). No relationship was discovered between size of design team and lay-familiarity with HCI.

Table 4.2 shows how design team size appears to be linked to involvement of an HCI specialist. In total 8 of the projects shown in the table involved at least one HCI specialist. Five others involved one or more individuals with some knowledge of HCI. Six out of the eight were projects over 60 person months (5 person years) long, and of these 4 were the four largest projects in the study.

This suggests that larger projects are more likely to involve HCI specialists. Projects with HCI specialists involved were compared for length against projects without, in a one-tailed T test (the hypothesis being that HCI specialist's projects were larger). The result was significant at the 0.005 level (0.5%) with \( t = 2.99 \), and 22 degrees of freedom. It seems safe to say that smaller design projects are less likely to have the resources to employ a specialist in HCI than larger projects.

An interesting finding was that two of the HCI specialists who responded to the questionnaire indicated that they did not have as much influence over the design as they would have wished. In one case the respondent described the fact that the design team ignored the weakness of the original design concept which led to the first usability test being very unsatisfactory. This respondent then had to engage in far more usability testing than should have been necessary. The other specialist was frustrated by the fact that usability testing seemed only to occur as a means of
resolving disputes amongst those involved in the design. He stated that the user
interface was repeatedly modified to suit the weaknesses in the application software,
rather than modifying the software to support a satisfactory UI design.

Table 4.4
Application Types for Twenty-Five Projects

<table>
<thead>
<tr>
<th>Text and File Oriented</th>
<th>No. of Projects</th>
<th>Group(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word Processing</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>Programming Environment</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>Window Manager</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>Office System</td>
<td>3</td>
<td>A BB</td>
</tr>
<tr>
<td>Communications</td>
<td>2</td>
<td>A B</td>
</tr>
<tr>
<td>Teaching</td>
<td>1</td>
<td>A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data &amp; Process Oriented</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Processing</td>
<td>8</td>
<td>AA BB CCC</td>
</tr>
<tr>
<td>Process Control</td>
<td>3</td>
<td>A B C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Design &amp; Problem Solving</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Aided Design</td>
<td>2</td>
<td>A B</td>
</tr>
<tr>
<td>Complex Problem Solving</td>
<td>1</td>
<td>B</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expert System</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Applied Expert System</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>Expert System Shell</td>
<td>1</td>
<td>A</td>
</tr>
</tbody>
</table>

In Group B all of the respondents were at least familiar with HCI (see table 4.1) even if their team did not include specialists in the field, this was only true of 42% of Group A and 20% of Group C. Bearing in mind that these were only small samples, the results seem to indicate an interesting difference between the informal classifications of the host organisations. Universities and R & D consultancies, who make up the bulk of this group, and (of course) HCI consultancies are more likely to employ HCI familiar people, or are more likely to carry out long term research
during which employees with a computer science background have a chance to educate themselves in more diverse fields.

An additional and interesting finding is that only seven (28%) of the respondents did not express a positive attitude towards HCI in theory. Some of these respondents were simply unaware of the discipline. However some of the respondents who were not themselves experienced in HCI, but were positive about it, may have been somewhat wary of the form in which it is typically presented. For example, in answer to the question "Could HCI be useful ?" two responses were "...not if books read like the standard AI text. Too academic." and "I would be very interested in the psychology of screen design, and ease of use if there is an easy to read text." Such responses tend to enforce the idea that it may be the form, rather than the content and aims of HCI guidelines and texts, which is the main problem with regard to its application.

4.3.4 Product Functions and the Type of UI involved

The questionnaire asked respondents to describe their design product's functions. Table 4.4 provides a rough classification of these responses into a number of groups. It is important to note the extreme diversity of these applications when compared with the limited range of applications studied using HCI DETs (see Chapters 2 and 3). Most notable here is that the applications for Group A and Group B do vary considerably, but all of the Group C responses belong in the Data and Process Oriented class of applications. Of the 5 in this group, 4 were data processing applications such as financial transactions, and one was a process control application (industrial parts movement). These applications were all part of the business of the design team's company.

Only one project was developing a system comparable with a text editor; this being a word processor, suggesting that the representativeness of text editors as applications on which to test HCI DETs may be questionable. One or two of the other applications such as one of the office systems (a multi author document preparation system) may have contained text editing facilities, but these would only represent a subset of the whole UI functionality. Two of the systems were communications oriented and might be comparable with CLG's EG system. However, it seems that the type of system typically used to test and demonstrate HCI DETs represents a
small minority of applications in general.

A variety of types of interface were described by the respondents. Most of these included menu style dialogues, but not all included a mouse. Many were described as WIMPs, some being direct manipulation. A few were forms based where users would fill in certain areas to give particular types of input to a database.

Common keywords used to describe the type of UI involved included:
- WIMPS
- WYSIWYG
- Menus
- Mouse
- Forms
- Function Keys

Of the 25 Uls described, at least 18 were menu-driven (identified by the keywords menu, mouse, WIMPS or Object Oriented), only 3 of these used cursor keys to select menu items. It seems therefore that "mousing" with the user selecting visible menu items is a more representative description of typical interaction styles than is command-driven input.

4.3.5 The Prospective Users

Respondents were asked to describe their system's prospective users. What is of interest here is not the actual nature or professions of users, although they turned out to range from children to garage staff. The real point of this question was to determine whether or not the design team were aware that their potential user population might be very different from themselves in terms of computer experience. This knowledge might impinge upon the value attributed to prospective user evaluations. All of the respondents were able to name some target group which was more specific than "the general public" for example.

Table 4.5 shows 25 user group descriptions based directly on the words of the respondents from the 25 projects. The user groups are informally classified as likely to be computer experienced or not. The vast majority of user groups described would seem to be likely to contain significant numbers of computer naive users.
In up to 20 out of the 25 projects described the designers would not have been able to rely on their prospective users being computer experienced; meaning having frequent access to and skills related to computers. The engineer designers group may have been computer skilled, however the respondent did not indicate the type of engineer involved. Different users described by respondents are listed in table 4.5; some teams shared prospective user groups; some mentioned more than one. These are categorised as those who might possibly be computer naive (one could not reasonably guarantee that they would know anything about computers, possibly having had no experience with them at all) and those who most probably will be computer experienced (i.e. are highly likely to have experience of use of a variety of systems for a number of tasks).

If users are very likely to be computer experienced then it may be safe to assume that they are familiar with many of the features of systems which systems designers consider to be natural; and that maybe the design team itself is fairly representative of prospective users. If there is a reasonable possibility that users are computer naive then the design team should at least carry out user evaluations to ensure that intuitive guesses at what is usable are actually correct.

For the large majority of prospective user groups described for the 25 projects in this study it is probably not reasonable to assume that they are familiar with computers and their conventions, although the designer engineers are perhaps the least likely to be in this position. It would also be unreasonable to expect many of these users to become highly skilled, since use of computers may only be a peripheral part of their job, and they may not receive any training or enough practice to gain sophisticated skills. Some design teams mentioned more than one type of user, in these cases designers need to assume that, as a whole, the most naive prospective users must be catered for as well as computer literate ones. Only four projects might reasonably have assumed that their prospective users were relative computer experts, on the basis that chip designers, unix and graphics workstation users and software engineers are likely to be professional computer scientists. Even in these cases user validations could be necessary to determine whether this is indeed true and the ease with which target tasks of these individuals might be carried out using the system.
Table 4.5
Potential Users Described by Respondents

<table>
<thead>
<tr>
<th>Computer Naive</th>
<th>Computer Experienced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Air Crew Selectors</td>
<td>Chip Designers</td>
</tr>
<tr>
<td>Office Staff</td>
<td>Unix Workstation Users</td>
</tr>
<tr>
<td>Office Workers</td>
<td>Graphics Workstation Users (two projects)</td>
</tr>
<tr>
<td>Garage Staff</td>
<td>Software Engineers</td>
</tr>
<tr>
<td>Computer Naive Domain Experts</td>
<td></td>
</tr>
<tr>
<td>Non Technical</td>
<td></td>
</tr>
<tr>
<td>Production Controllers</td>
<td></td>
</tr>
<tr>
<td>Financial Planners</td>
<td></td>
</tr>
<tr>
<td>Financial Database Users</td>
<td></td>
</tr>
<tr>
<td>Authors</td>
<td></td>
</tr>
<tr>
<td>Computer Naive</td>
<td></td>
</tr>
<tr>
<td>Secretaries (two projects)</td>
<td></td>
</tr>
<tr>
<td>Gallery Staff</td>
<td></td>
</tr>
<tr>
<td>Data Entry Clerks</td>
<td></td>
</tr>
<tr>
<td>Estimators &amp; Maintenance</td>
<td></td>
</tr>
<tr>
<td>Clerks</td>
<td></td>
</tr>
<tr>
<td>Telephone Reps</td>
<td></td>
</tr>
<tr>
<td>Children &amp; Schoolteachers</td>
<td></td>
</tr>
<tr>
<td>(Engineer Designers; perhaps)</td>
<td></td>
</tr>
</tbody>
</table>

4.3.6 Organisations involved

Designers were asked to describe other organisations (i.e. client or marketer) involved with the project, and the extent of their involvement. Groups A and B described a number of different arrangements, with the two largest Group B projects being described as being funded by Alvey (a government IT initiative for funding collaborative research by industrial and academic organisations). The Group C respondents all described their projects as being internal to their company as expected.
Of the 12 design projects belonging to Group A, seven products were marketed at general target groups by the design team’s company itself. Three were projects undertaken for one client organisation (and in one case two) only. One was sold through a marketing organisation, and another was sponsored by the government for schools, representatives of which collaborated with the team.

Of the 8 Group B design products, three were designed for one client organisation, two were sold via marketing organisations, two were government funded research projects, and one was aimed at a general target group (the academic community) representatives of which were close at hand and able to collaborate in the design process.

In Group C three projects were within the design team’s own department, and two involved collaboration with another department (or two departments)

In total there were fifteen Group A, B and C projects which involved collaboration with individuals in a client organisation, collaborating organisation or another company department. Only three respondents stated that the collaboration or client assistance was detrimental to the project. Notably one respondent complained that the physical distance between collaborators on an Alvey project made the work very difficult. However in most cases collaborator/clients proved to be a valuable resource for the design team. They supplied equipment, information, expertise, and test beds, often with potential users.

4.3.7 Information from Requirements, and Design Specifications, Task Descriptions and Users

The information obtained from this part of the questionnaire was diverse and difficult to classify. However the responses in the later user-oriented information sources provide a clearer picture of this information. Here a more qualitative summary is given of the responses, based upon 25 different projects.

Six questions were devoted to finding out what information was used to drive the design. Seventeen respondents used some kind of requirements or design (functional) specification for the UI. Only three of these specifications were described as formal, the rest were in "English" and some were described as being very informal
or brief. As an illustration, one respondent returned all of the general requirements and early design specification material from the project. This comprised about 30 pages for the whole system of which only one page was devoted to user requirements and UI task oriented functionality. Requirements and design specifications would usually be agreed with a client or collaborator if there was one. One respondent stated that the design specification was made up as the design went along, this seems to be more like documentation. At the opposite extreme, another respondent stated that an early design specification was produced which was used throughout to drive the design and compare with its outcome.

Three respondents used particular specification methods. Two stated that in-house specification techniques were used, and one that VDM, which is a strictly formal specification technique, was used. One of the in-house techniques mentioned was described as a "formal" requirements specification technique, the other was not defined. It is not clear whether the latter, in-house method used was actually a formal method like VDM, a structured system specification method, or a requirements specification technique, however a later response from this project suggests that a formal approach to generation and testing was used. The exact meaning of the term formal to describe an approach may well be variable in software engineering as well as in other domains.

There was some evidence which suggested that abstract design specifications are not considered practical by some designers. Two quotes were highly illustrative; "There never was time to indulge in such luxuries (the management claimed...)", and "None [were used]. This is the real world!"

Information about to-be-supported target tasks was gathered in a number of ways. Written specifications of target tasks were sometimes supplied by a client or collaborator. Task analyses, verbal task descriptions from potential users or management, observation of prospective users, documentation on related tasks were all used on more than one project. The design team themselves might write the target task specification and agree on it with the client. Sometimes it was a joint activity. More detail about information sources came from part II of the questionnaire.

What was most disturbing was that eight of the respondents relied very heavily on indirect information supplied by managers, or documentation, without finding out
more about the actual needs of the users themselves. In 5 of these cases, representing 20% of the total number of projects in this study, it was because the user population was difficult to get hold of or the tasks were novel. The other three, simply did not involve users until the evaluation stages. Another design team included representative potential users and for this reason only used other potential users for evaluation.

Information from users, where it was used for the design, as opposed to evaluation of the UI, seems to have come largely from informal, unobtrusive observations, interviews and verbal task descriptions. UIs were often tested internally, in the design team’s own organisation, and then outside, on a client’s site, or with prospective users elsewhere.

Where users took part in evaluations, they generally used the system prototype, rather than the completed design product. Informal, and some formal experiments were common, with users attempting specified tasks using the prototype. In general, prospective users appear to play a very important role in the majority of design projects, though they seem to be involved much more readily in evaluation than design, as one respondent put it "They were better at reacting than initiating". In 4 teams overall, no user evaluations took place. In one case this seemed to be because the design team felt that they were sufficiently representative of users. In the other 3 cases, representing 12% of the projects studied, where users would be financial planners, secretaries, and computer naive users, there was no good explanation for this lapse.

One respondent stated that "Though the initial concept was poor the design team went ahead (most subsequent problems arise from this)...After the first exploratory usability test, the design team acknowledged the problems and started redesign aimed at the second release." This quote illustrates the value of early evaluations or user-oriented design which could have saved the first release of the product from being so problematic.
Table 4.6
Structured Specification Techniques and Descriptive Methodologies
Used for Requirements, Design and Evaluation

<table>
<thead>
<tr>
<th>Name as Specified</th>
<th>Assumed Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Analysis (3 cases)</td>
<td>Any general method for categorising, breaking down, generifying, identifying goal structures, rules etc involved in tasks.</td>
</tr>
<tr>
<td>STNs (2 cases)</td>
<td>State Transition Networks, representing states and actions in a system.</td>
</tr>
<tr>
<td>User Conceptual Design</td>
<td>A structure which reflects or simulates the desired user’s view of the system states and actions. It must include the entities and operations involved in all target tasks.</td>
</tr>
<tr>
<td>Adapted VDM</td>
<td>Vienna Definition Methodology is a formal specification technique developed by IBM with which it is possible to prove for given sets of conditions, given functions will obtain true. It is not generally used for UI design in particular.</td>
</tr>
</tbody>
</table>
Table 4.6 Continued
Structured Specification Techniques and Descriptive Methodologies
Used for Requirements, Design and Evaluation

<table>
<thead>
<tr>
<th>Name as Specified</th>
<th>Assumed Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSD</td>
<td>Jackson's Structured Design methodology is a systems analysis and design technique. This methodology in its popular form includes little support for user-oriented design.</td>
</tr>
<tr>
<td>ART</td>
<td>A knowledge engineering environment. Supports multiple paradigm based modelling of knowledge (e.g. production-rule-based, frames-based etc).</td>
</tr>
<tr>
<td>MAID</td>
<td>Unspecified</td>
</tr>
<tr>
<td>BNF</td>
<td>A program specification language, not a true design methodology.</td>
</tr>
<tr>
<td>DFDs</td>
<td>Data Flow Diagrams A graphical way of modelling data flow and transformations in a system.</td>
</tr>
<tr>
<td>Internal Design Methodology</td>
<td>Mentioned but not specified by one respondent.</td>
</tr>
</tbody>
</table>
When asked how familiar they were with the application domain of their system, designers varied unpredictably in their responses. The group, application, type of user was not related to design team's knowledge about the tasks, problems and so on associated with a particular domain. The hypothesis that Group C respondents would be more familiar with their application was not supported. Two out of the 5 design teams in this group were unfamiliar with the domain, and the other three were partially familiar in two cases because half of the design team were application experts whereas the other half knew little about it.

Only about a third of the respondents stated that they were very unfamiliar with the application domain, some of these respondents were members of teams which involved application experts who might make up for the problems of unfamiliarity. Where familiarity was low design teams generally seemed to spend time familiarising themselves with the application, either by watching current or manual systems users, or by talking to applications experts. This kind of information supplemented requirements specifications, and target task descriptions.

4.3.8 Structured Specification Techniques and Descriptive Methodologies Used for Requirements, Design and Evaluation

Respondents, having described what information was used were then asked to state what abstract design specifications, or particular descriptive methodologies they used. Some common examples were used to indicate the type of technique implied, including Task Analysis, Interaction Grammar, JSD, GOMS, LARCH). The acronyms were not explained, since designers who had not used them might have classified their own informal procedure as being one of these if they felt it was similar. What was of interest here was not the scope of the techniques suggested by the two questions relating to this issue (which were left as general as possible) but the type of techniques reported, and their prevalence. The questionnaire stated twice that it was the UI design which was the main focus of the system, not the general functionality of the application software. However, as table 4.6 shows, the majority of structured techniques reported were not user-oriented.

Thirteen projects used structured techniques of some sort. Ten respondents claimed to have used an abstract design specification, or descriptive methodologies. Of these, six mentioned one approach and four used two different approaches. Ten
different approaches were identified overall, these are shown in table 4.6. The main feature here is that many of the specification and descriptive techniques used are not user-oriented. Task Analysis, and the User Conceptual Design, are clearly aimed at specifying the UI behaviour with respect to user characteristics. With five other techniques (STNs, VDM, JSD, BNF, DFDs) which are not user-oriented, it may be as easy to specify an unusable system as a usable one. These techniques are really aimed at improving design of system functionality so that it is elegant, efficient, reliable, and/or modifiable. These aspects of the system are not sufficient to ensure usability.

Only 4 projects out of 25 used early user-oriented design abstractions or specifications. These were the task analysis techniques and the user conceptual design methodology shown in table 4.6. This suggests that projects which attempt to build an early model of users or their tasks which could be used to drive the design are in a minority. Other projects may have relied more on evaluation to get this kind of information.

Perhaps most important of all the findings in this study is the fact that not one project mentioned use of an HCI DET of the type described in Chapter 2. Whether or not any of the projects were really rigorously user-oriented at all was hard to discern from the responses.

4.3.9 Generation and Evaluation

When asked to describe how the UI was generated and evaluated for usability, a very wide range of responses were given. A number of keywords or ideas were used by respondents to describe their approach to generation and testing. These are listed in table 4.7, (note that some projects used more than one approach). A more detailed inspection of responses revealed that the term formal was sometimes used (in three instances) to refer to any procedure which was in some way structured, such as an experiment. In the other six cases the term may have meant either mathematically formal or rigorous.

The number of respondents who used these terms or ideas in their responses may only be a subset of those who actually could have done. The only clear suggestion of these responses is that informal, iterative methods seem to be far more common
Table 4.7
Keywords used to describe Generation and Testing Methods

<table>
<thead>
<tr>
<th>Keyword/Idea</th>
<th>Number of Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reporting Use</td>
</tr>
<tr>
<td></td>
<td>(and Percentage of Total)</td>
</tr>
<tr>
<td>Formal</td>
<td>9 (36%)</td>
</tr>
<tr>
<td>Test-against-specs</td>
<td>5 (20%)</td>
</tr>
<tr>
<td>Informal</td>
<td>13 (52%)</td>
</tr>
<tr>
<td>Iteration or Prototyping</td>
<td>14 (56%)</td>
</tr>
</tbody>
</table>

than formal, functional/requirements specification-based generation and testing approaches (this contrasts with the findings of Rosson et al 1987; where the split between phased and incremental, iterative design was more even).

Clients, prospective users, non-prospective users, members of the design team, and external consultants were all variously mentioned by one or more of the respondents as being involved in the generation, testing, and usability evaluations of the system. Most of the respondents mentioned that they tested their systems themselves, usually before going on to carry out more extensive testing and evaluations.

Usability evaluation of the prototype or final product varied extensively as shown in table 4.8 where each project is represented by only a single method; this is possible since only those projects using early evaluation listed more than one method. In three instances there were no user evaluations. In another only feedback from demonstrations was used. In at least seven cases usability evaluation was delayed until late in the design cycle. At least seven were able to use iteration as a device for incorporating user feedback and another was left to evolve as requirements grew after implementation. Six respondents described early user evaluations, using such techniques as mock-up simulations, concept tests, and user involvement from initial design stages. These projects also used other methods later on in design. Of these
6, 4 were groups that also used abstract design specifications, or descriptive methodologies, suggesting a more generally rigorous approach to their design.

**Table 4.8**
Varied Methods of Evaluation in Projects Studied

<table>
<thead>
<tr>
<th>Evaluation</th>
<th>Number of Projects (and Percentage of Total)</th>
</tr>
</thead>
<tbody>
<tr>
<td>None Existent</td>
<td>3 (12%)</td>
</tr>
<tr>
<td>Demonstration Based</td>
<td>1 (4%)</td>
</tr>
<tr>
<td>Late</td>
<td>7 (28%)</td>
</tr>
<tr>
<td>Late Evolutionary Iteration</td>
<td>1 (4%)</td>
</tr>
<tr>
<td>Iterative</td>
<td>7 (28%)</td>
</tr>
<tr>
<td>Early (&amp; other methods)</td>
<td>6 (24%)</td>
</tr>
</tbody>
</table>

In general, from table 4.8, it appears that there is a bias in design practice towards informal generation and testing, and towards later, informal evaluations, rather than towards more rigorous user-oriented approaches with early specifications and evaluations.

**4.3.10 Problems and Modifications**

Respondents were asked whether they had experienced any notable problems on their projects or had to make any notable modifications. Three stated that no major problems or modifications were experienced. In total six stated that no notable problems emerged, and twelve that no notable modifications were required. This means that in 22 projects notable problems did emerge, and in 11, major alterations had to be made because of these problems. Another two projects made significant modifications for other reasons.
Table 4.9
Notable Problems and Modifications Experienced by Respondents

<table>
<thead>
<tr>
<th>Project &amp; Problem(s)</th>
<th>Modification(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1A None</td>
<td>None</td>
</tr>
<tr>
<td>2A Slowness &amp; Complexity of part of the application.</td>
<td>Simplification of UI functionality to make it understandable but less comprehensive.</td>
</tr>
<tr>
<td>3A Problems caused by designers ignoring weakness in initial concept.</td>
<td>After first usability test designers acknowledged problems and began redesign aimed at the second release.</td>
</tr>
<tr>
<td>4A Designer accidentally discovered a frequent action required several menu choices.</td>
<td>Added a direct option for the frequent action.</td>
</tr>
<tr>
<td>5A None</td>
<td>None</td>
</tr>
<tr>
<td>6A Ad-hoc design with little regard given selected machine capability, with the UI being modified to suit the implementation.</td>
<td>Almost every time a module reached a significant milestone in its design.</td>
</tr>
<tr>
<td>7A Technical limitations of other equipment the team's package interfaces with.</td>
<td>None</td>
</tr>
<tr>
<td>8A Upgrades in the operating system proved detrimental to system responses.</td>
<td>Upgrade in hardware to counteract slow response times.</td>
</tr>
<tr>
<td>Project &amp; Problem(s)</td>
<td>Modification(s)</td>
</tr>
<tr>
<td>----------------------------------------------------------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td>9A System too slow due to specific device drivers.</td>
<td>None</td>
</tr>
<tr>
<td>10A Lack of personnel; lack of target machine power; general lack of foresight and investment.</td>
<td>None</td>
</tr>
<tr>
<td>11A Managing finite hardware resources and getting hardware to work.</td>
<td>None</td>
</tr>
<tr>
<td>12A Representation of language accents.</td>
<td>Defined accented characters using function keys.</td>
</tr>
<tr>
<td>1B No methodology available for the application; chose the wrong language.</td>
<td>Changed to Smalltalk.</td>
</tr>
<tr>
<td>2B Bad management &amp; programmer; nebulous deadlines &amp; contractual problems; expense.</td>
<td>Scrapped prototype &amp; prototyper. Only minor changes made to spec.</td>
</tr>
<tr>
<td>3B None</td>
<td>UI simplified due to feedback from demo’s.</td>
</tr>
<tr>
<td>4B Underestimated system capabilities. Better prototyping may have helped.</td>
<td>Included more dynamic feedback &amp; direct manipulation.</td>
</tr>
</tbody>
</table>
Table 4.9 Continued
Notable Problems and Modifications Experienced by Respondents

<table>
<thead>
<tr>
<th>Project &amp; Problem(s)</th>
<th>Modification(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5B None</td>
<td>Rewrote graphics interface late in project to allow porting to Sun workstations.</td>
</tr>
<tr>
<td>6B French English translations.</td>
<td>None</td>
</tr>
<tr>
<td>7B Screen handling library software was poor; bad documentation &amp; tags.</td>
<td>None</td>
</tr>
<tr>
<td>8B Collaborators too far away.</td>
<td>None</td>
</tr>
<tr>
<td>1C Technological limitations of screens; even though better and cheaper were available.</td>
<td>None</td>
</tr>
<tr>
<td>2C Hard to get users to agree on detailed requirements. This should have been tackled earlier in the projects.</td>
<td>None</td>
</tr>
<tr>
<td>3C None</td>
<td>None</td>
</tr>
<tr>
<td>4C System didn't do exactly what clients wanted. perhaps prototyping would have helped.</td>
<td>Heavy maintenance in first year to correct design deficiencies &amp; fix bugs.</td>
</tr>
<tr>
<td>5C None</td>
<td>Changes in page breaks.</td>
</tr>
</tbody>
</table>
Table 4.9 shows notable problems and modifications described for the 25 projects. The amount of variability in these responses indicates, once more the variability in the whole process of design. The problem responses can be classified into a number of groups see below. Two projects, 2A & 1C, experienced problems which appear under two classifications. Note that the projects are referred to by their host organisation classification group (see table 4.1) and a number (purely for the purpose of distinguishing class members).

**Technological problems experienced by projects:** 2A, 7A, 8A, 9A, 11A, 12A, 7B, 1C

**UI Complexity experienced by projects:** 2A, 4A

**Poor design concept experienced by projects:** 3A, 6A

**Organisational experienced by projects:** 10A, 2B, 8B, 1C

**Methodological experienced by projects:** 1B, 4B, 2C, 4C

**Pragmatic experienced by projects:** 6B

This classification scheme is summed up in table 4.10, which is essentially a condensation of table 4.9. These problems were not related to whether or not the project was large or the familiarity of the design team with HCI. Table 4.10 shows the incidence of these classes of problems for each of the three groups of organisation identified here. Technological and organisational problems seem to be more apparent than design concept and usability based problems. It may be that these problems are simply the ones which are easier for most designers to identify. The most interesting finding is that half of Group A experienced technological problems.

A Chi squared test was carried out with two expected cell values of 4, which were compared with the observed values. A value for Chi of 9 was achieved which with the one degree of freedom for this test, is highly significant at the 0.5% level (P = 0.005).

This extremely high degree of statistical significance tends to balance the smallness of the sample (which would normally be advised against for a Chi squared test; Robson 1973) and suggests that Group A is significantly more prone to technological problems than Groups B and C. Unfortunately the scope of this study does not provide an explanation for this phenomenon, however as an anecdote, it is worth noting that personal experience of the author suggests that computer software houses and lone consultants are usually less well equipped with hardware and software than university and R & D organisations' staff.
An important point to make here is that design problems should be considered as representing design constraints in themselves. Some of these problems may be avoidable and it is these which should be dealt with by HCI techniques, whereas the unavoidable should be considered as part of the overall character of the design process within which HCI approaches must be able to work.

Table 4.10

Classification of 25 Design Teams According to Host Organisation and Incidence of Types of Design Problem

<table>
<thead>
<tr>
<th>Type of Problem</th>
<th>Group A</th>
<th>Group B</th>
<th>Group C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Software Companies &amp; Consultants</td>
<td>R &amp; D Groups, Universities &amp; HCI Consultancies</td>
<td>Departments of Non Computer Companies</td>
</tr>
<tr>
<td>Number in Study</td>
<td>12</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>Technical</td>
<td>6</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Organisational</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Methodological</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>UI Complexity</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Poor design concept</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Pragmatic</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>None</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

4.3.11 Finalisation of the Project

Designers were asked to describe how their product was finalised. Three projects were incomplete, including the project with two respondents from different collaborators, and one respondent did not answer, leaving 21 responses to this question. Table 4.11 shows that eight responses indicate that an agreement was reached, or an
earlier specification (by agreement) matched by the product. Another seven described a process of iteration towards a finalised product; which implies some agreement as to when to discontinue iteration.

Two projects simply ran out of time, one of these overran anyway. Another could not improve the design because of limitations in its technological resources. One was terminated by the team leader's decision, and the last project was transferred to another site. The three responses stating that resources were exhausted may represent cases where the whole team would have liked to improve the design.

Table 4.11
Finalisation of Project

<table>
<thead>
<tr>
<th>Manner of Finalisation</th>
<th>Number of Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agreement</td>
<td>7</td>
</tr>
<tr>
<td>Specification Satisfied</td>
<td>1</td>
</tr>
<tr>
<td>Iteration</td>
<td>7</td>
</tr>
<tr>
<td>Resource Exhaustion</td>
<td>3</td>
</tr>
<tr>
<td>Team Leader Decision</td>
<td>1</td>
</tr>
<tr>
<td>Informally Finalised</td>
<td>1</td>
</tr>
<tr>
<td>Project Transferred</td>
<td>1</td>
</tr>
</tbody>
</table>

The questionnaire asked if the respondent was satisfied with the final product. Six respondents (28% of the completed projects) stated that they were not. Thirteen gave an unqualified assertion that they thought the product was good, and 3 used qualifications such as "Yes, given the constraints of the environment." Surprisingly there was no apparent link between satisfaction and the manner in which the project was terminated.

Interestingly, of the thirteen who gave unqualified Yes answers to this question, three had not carried out user evaluations, and five had only carried out late evaluations. In fact this group, as well as being the most common response (62% of the completed projects), also represented the majority (eight) of the eleven projects which had late or non existent user evaluations. There does not seem to be any
relation between rigour of usability evaluation and satisfactoriness of the design product as perceived by the respondent members of the design teams. This is perhaps a truism, since those who do not look for problems in their design are unlikely to perceive them.

4.3.12 Lessons Learned

Finally designers were asked to state what, if anything, they would have done differently with the benefit of hind sight. Fifteen respondents stated that they would have done something differently. Many of these responses revolved around choice of hardware or software. Those that did not relate to hardware or software included diverse comments such as "I would have added more features to make the client think", "I would not have collaborated with a group of people 100 miles away", "I would have taken someone else's design from day one", and "I would have run the project with more commercial acumen".

Six of the responses to this question suggested that the respondent would have taken a different methodological approach altogether. For comparison with table 4.9, table 4.12 identifies the project respondents who would have taken a different approach and the comments they made.

These comments mostly relate to the notable problems which emerged in the projects. 3A and 6A (both members of the software company/consultancy group) are responses from HCI specialists whose questionnaire responses indicated that they did not feel they had enough influence in the design process, and what influence they did have was too late. 12A was not familiar with HCI but obviously positive about it and took a meticulous approach throughout using extensive, rigorous and iterative usability testing. However, this respondent used no early task specifications or evaluations to help drive the design. 1B had the problem of being unable to find an appropriate design methodology, and ended up changing over to object-oriented software which is considered to be better suited to prototyping, modification and extension than other types of software, (Bullinger et al, 1984; Booch, 1986; Meyer, 1988). 4B and 4C, both had problems with a lack of prototyping. 4B underestimated system capabilities and the respondent seems to think that an earlier, more specific design would have been helpful. 4C ended up with a product which did not completely satisfy the client, and the respondent seems to blame
Table 4.12
Projects Where Different Approaches Were Indicated
With the Benefit of Hindsight

<table>
<thead>
<tr>
<th>Project</th>
<th>Comment on Different Approach Desired</th>
</tr>
</thead>
<tbody>
<tr>
<td>3A</td>
<td>&quot;I would have fought harder for different emphasis and direction at the start.&quot;</td>
</tr>
<tr>
<td>6A</td>
<td>&quot;I would have used expert knowledge elicitation to define functionality and would base the design around this (i.e. reverse the roles).&quot;</td>
</tr>
<tr>
<td>12A</td>
<td>&quot;I would have preferred a more rigorous specification, but this would have been hard to achieve.&quot;</td>
</tr>
<tr>
<td>1B</td>
<td>&quot;I would have started with an appropriate methodology (a designed one).&quot;</td>
</tr>
<tr>
<td>4B</td>
<td>&quot;I would have used a more specific design earlier.&quot;</td>
</tr>
<tr>
<td>4C</td>
<td>&quot;I would have prototyped - no convenient methods existed at the time.&quot;</td>
</tr>
</tbody>
</table>

this on the inability to prototype at the time of the project.

4.3.13 Constraints on Design Activity

The second part of the questionnaire involved two ranking exercises. The first required respondents to read a list of possible design constraints. A design constraint may be anything which impedes or prevents a design team from progressing
in what they perceive to be an ideal manner towards their design goals. It may increase the difficulty of carrying out certain activities, or it may prevent certain activities taking place. The constraints listed below are a mixture which could be placed into the categories indicated by Hammond et al (1983) which are Historical, Organisational, System and Personal (see figure 3.3). They were derived from several pilot conversations which the analyst had with some individuals with commercial design experience at Queen Mary College Computer Science Department. The constraints listed are as follows:

a) Lack of Autonomy From Parties Outside of Design Team
b) Lack of Guidance From Parties Outside of Team
c) Lack of Authority
d) Oversized Team
e) Undersized Team
f) Undefined Team Member Roles
g) Over-rigid Team Member Roles
h) Lack of Assistance/Collaboration from Client
i) Client Over-Intervention
j) Lack of Information about Tasks
k) Lack of Information about Users
l) Over-casual Approach to Design
m) Over-rigid Approach to Design
n) Over-casual Approach to Evaluation
o) Over Rigid Approach to Evaluation
p) Lack of Experience With HCI
q) Lack of Experience with Interface Design
r) Lack of Information About What Constitutes Interface Design Improvement
s) Lack of Familiarity of Application Domain
t) Complicated Application/ Sophistication of Product
u) Inadequate Resources (e.g. Time, Money, Equipment etc)

The rationale for including these items in the list is, briefly, as follows. Items a and b are aimed at discovering whether designers feel they have too little or too much freedom. Item c asks how much power designers have in asserting their decisions. For d, e, f, and g the aim is to determine whether size and degree of team structure
or lack of it might be a problem. Items h and i are to determine whether clients of
design teams are helpful or constraining in themselves. The availability of user and
task information is queried by j and k. Items l, m, n, and o seek to determine
whether design and evaluation is perceived to be too informal or too rigid by
designers. The designers' familiarity with HCI, UI design, ways of improving usa-
bility, and with the application domain itself is queried by p, q, r, and s. Finally the
complexity of the application/sophistication of the product, which might be hard to
understand and require time and effort simply to specify, is queried by t. Item u
aims to determined whether time and other resources are sufficient to achieve the
kind of results the design team would like.

Respondents were asked to choose which constraints they felt they had experienced
and then list them in order of their importance to the single design project which
they were referring their responses to. A valid response was therefore taken to be
any ordered list (including items placed as equal) of the constraints above. Respon-
dents provided a variety of rankings which were then combined to give overall
ranking scores for each of the possible constraints. In this section the two responses
from the same multi-site project are treated separately. The reason for this was that
they provided very different responses which indicated that their groups were
operating autonomously from one another. One respondent stated that his rankings
were based on general experience, rather than a single design project. This was
considered unlikely to be unrepresentative for a single project and this data was
therefore not included (virtually every option was included in the ranking by this
respondent). Another respondent did not complete this section. These respondents
were both in group C. In total there were therefore 24 responses.

Respondents varied enormously in terms of the number of constraints they listed.
These ranged from 1 to 21 (out of 22), however there was no discernible difference
between the groups, between long and short projects and between teams with and
without HCI specialists in terms of the number of constraints listed. One might
have assumed that commercial projects are more subject to constraints and pres-
sures, however this intuitive opinion is not supported by this study. Unfortunately
only 3 of the 5 Group C respondents gave valid responses for the ranking exercises,
and as such do not provide a large enough group to support a more detailed statisti-
cal analysis. The average number of constraints listed by members of Group A
(6.00) is similar to the average for Group B (6.5). This suggests that Universities, R
& D organisations, and HCI consultancies are under the same sorts of pressures as commercial computer software design companies and consultants.

Interestingly almost all of the respondents ticked many more constraints than design problems. It is possible that designers make subtle distinctions between what they consider to be actual problems, and what they consider to be simply constraints upon their activities. A constraint may be an unavoidable problem to which the designer sees no practical solution.

Two different rankings were carried out in the analysis of the constraints data, the first based upon "Incidence" or number of times each constraint was listed as experienced by respondents. So in this case only the number of times an individual constraint was listed (as having been experienced on a project) was the basis for its "Assigned Frequency Rank". The first ranking results are shown in table 4.13.

All of the suggested constraints in the list were existent in at least two projects. The most common constraint appears to be the Complexity of the Application/Sophistication of the Envisioned Product which seems to be almost twice as common as the next most common; Inadequate Resources.

In the second ranking analysis the top ten constraints were accepted from each respondent and assigned a "Weighting". The first item was scored at 10 points, the next at 9, and so on down to 1 point. These weighted rankings were added together (again one respondent's rankings were based on general experience and so were not included). The total score for each constraint was recorded; the "Overall Weighting". In this case, then, the rank of each constraint represents its overall importance; the "Assigned Importance Rank" (see table 4.14). Two respondents gave many more than ten ranked constraints (these few additional constraints were not counted), all of the other respondents gave up to ten.

In this case, the most important constraint is Complicated Application/ Sophistication of Product (indicating a complex and difficult system to build) but the second most important constraint is Lack of Information about Users, followed by Over-casual Approach to Evaluation.

Application complexity or product sophistication was originally viewed as a
Table 4.13
Incidence of Constraints Experienced by Respondents

<table>
<thead>
<tr>
<th>Item in List</th>
<th>Incidence</th>
<th>Assigned Frequency Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>t) Complicated Application/Product Sophistication</td>
<td>19</td>
<td>1</td>
</tr>
<tr>
<td>u) Inadequate Resources</td>
<td>11</td>
<td>2</td>
</tr>
<tr>
<td>k) Lack of Information about Users</td>
<td>10</td>
<td>3.5</td>
</tr>
<tr>
<td>n) Over-casual Approach to Evaluation</td>
<td>10</td>
<td>3.5</td>
</tr>
<tr>
<td>e) Undersized Team</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>h) Lack of Assistance/Collaboration from Client</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>q) Lack of Experience with Interface Design</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>r) Lack of Info About UI Design Improvement</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>b) Lack of Guidance From Parties Outside D-Team</td>
<td>7</td>
<td>10.5</td>
</tr>
<tr>
<td>f) Undefined Team Member Roles</td>
<td>7</td>
<td>10.5</td>
</tr>
<tr>
<td>j) Lack of Information about Tasks</td>
<td>7</td>
<td>10.5</td>
</tr>
<tr>
<td>p) Lack of Experience With HCI</td>
<td>7</td>
<td>10.5</td>
</tr>
<tr>
<td>c) Lack of Authority</td>
<td>6</td>
<td>13.5</td>
</tr>
<tr>
<td>l) Over-casual Approach to Design</td>
<td>6</td>
<td>13.5</td>
</tr>
<tr>
<td>s) Lack of Familiarity of Application Domain</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>i) Client Over-Intervention</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>d) Oversized Team</td>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>g) Over-rigid Team Member Roles</td>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>o) Over Rigid Approach to Evaluation</td>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>a) Lack of Autonomy From Parties Outside D-Team</td>
<td>2</td>
<td>20.5</td>
</tr>
<tr>
<td>m) Over-rigid Approach to Design</td>
<td>2</td>
<td>20.5</td>
</tr>
</tbody>
</table>

candidate design constraint because it would be more likely to give rise to complex problems which are harder to reason about than simple ones (e.g. Newell & Simon, 1972). This being the case, designers in complex application projects could be likely to spend a greater proportion of their time problem solving; perhaps finding any solution at all to each complex problem might be the design goal, rather than
selecting the best solution. In such a situation less time will be available for reviewing alternative solutions, or being creative.

Comparing the two ranking schemes in tables 4.13 and 4.14, it seems that there are few major differences between them, although the differences that do exist illustrate

<table>
<thead>
<tr>
<th>Item in List</th>
<th>Overall Weighting</th>
<th>Assigned Importance Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>t) Complicated Application/Product Sophistication</td>
<td>149</td>
<td>1</td>
</tr>
<tr>
<td>k) Lack of Information about Users</td>
<td>74</td>
<td>2.5</td>
</tr>
<tr>
<td>n) Over-casual Approach to Evaluation</td>
<td>74</td>
<td>2.5</td>
</tr>
<tr>
<td>u) Inadequate Resources</td>
<td>71</td>
<td>4</td>
</tr>
<tr>
<td>h) Lack of Assistance/Collaboration from Client</td>
<td>60</td>
<td>5.5</td>
</tr>
<tr>
<td>q) Lack of Experience with Interface Design</td>
<td>60</td>
<td>5.5</td>
</tr>
<tr>
<td>e) Undersized Team</td>
<td>48</td>
<td>7</td>
</tr>
<tr>
<td>b) Lack of Guidance From Parties Outside of Team</td>
<td>46</td>
<td>8</td>
</tr>
<tr>
<td>p) Lack of Experience With HCI</td>
<td>40</td>
<td>9</td>
</tr>
<tr>
<td>l) Over-casual Approach to Design</td>
<td>38</td>
<td>10</td>
</tr>
<tr>
<td>c) Lack of Authority</td>
<td>37</td>
<td>12.5</td>
</tr>
<tr>
<td>f) Undefined Team Member Roles</td>
<td>37</td>
<td>12.5</td>
</tr>
<tr>
<td>j) Lack of Information about Tasks</td>
<td>37</td>
<td>12.5</td>
</tr>
<tr>
<td>r) Lack of Info About UI Design Improvement</td>
<td>37</td>
<td>12.5</td>
</tr>
<tr>
<td>s) Lack of Familiarity of Application Domain</td>
<td>24</td>
<td>15</td>
</tr>
<tr>
<td>i) Client Over-Intervention</td>
<td>23</td>
<td>16</td>
</tr>
<tr>
<td>a) Lack of Autonomy From Parties Outside D-Team</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>d) Oversized Team</td>
<td>10</td>
<td>18</td>
</tr>
<tr>
<td>o) Over Rigid Approach to Evaluation</td>
<td>9</td>
<td>19</td>
</tr>
<tr>
<td>m) Over-rigid Approach to Design</td>
<td>8</td>
<td>20</td>
</tr>
<tr>
<td>g) Over-rigid Team Member Roles</td>
<td>3</td>
<td>21</td>
</tr>
</tbody>
</table>
that some constraints are perhaps more common but less important than others and vice versa.

Both of the rankings suggest that there are less complaints about rigid design approaches, and team roles than about the lack of structure in the design method, approach goals and organisation which seems to be typical of the mainly informal design approaches indicated in this study. One designer provided an illustrative comment "some prototyping iterations had no clear objectives", suggesting that design teams may go through the motions of generating and evaluating their designs without having a clear idea of what they are aiming for. There may be some support here for the idea that structured design methods of the right kind could be a welcome improvement for design teams. The problem is finding the right methods.

Furthermore the value of the assistance of clients in providing information about tasks, users and requirements, resources and expertise etc tends to be confirmed by the rankings. Even though many of the teams did not collaborate with clients, those that did seemed to think that more assistance would have improved the design, to such an extent that Lack of Assistance from Client was ranked as the sixth most common and fifth most important design constraint.

Respondents were invited to add any other constraints which they experienced but which were not themselves on the list. These were as follows, listed with the identity of the project from which they were drawn;

1A Over zealous project manager.
2A Time.
   Early purchase of lastest technology.
   Parallelism of development activities.
3A Lack of high quality investigation information and competitor awareness to start.
5A Time - more work in whole project (not just UI) than anticipated - negotiated with client to reduce functionality.
7A Lack of time.
8A Software restrictions.
   Hardware restrictions.
9A Machine limitations (memory and speed).
Portability of UI.

10A Lack of foresight and liaison with hardware team.
11A Lack of time.
1B Lack of time.
2B Worker attitude.
3B Incompetence of assisting staff.
   Difficulty of motivating collaborators when required.
   Disruption moving to new building.
7B Lack of experience of users.
   Lack of suitable programming environment.
   Slow hardware.
4C Time required under estimated.
   Difficulty debugging programs.

These additional constraints were all ranked by the respondents as first, second or third most important (apart from the disruption moving to a new building which was ranked 5th by 3B). The comments suggest that respondents felt that time and resources in themselves are very important as design constraints (e.g. 2A, 8A, and 9A), even though they were listed in the questionnaire under Inadequate Resources see Appendix 1. The majority of other major problems not included in the questionnaire items seem to be concerned with criticisms of project members (e.g. 1A 10A, 2B, and 3B), however, these seem to be unlikely to be avoided by adopting other design strategies.

4.3.14 Information Sources

The same ranking exercises as described for constraints (Assigned Frequency Ranking and Assigned Importance Ranking) were carried out on user-oriented information sources which the respondent’s design team might have exploited. Information sources included in the list are shown below. These sources vary in terms of whether they can be used to generate design or to evaluate it, or both. As with design constraints there was a great deal of variation between design projects as to the number of information sources exploited. Twenty-five valid responses were obtained for this exercise. Again, as for the constraints, a valid response was taken to be any ordered list of information sources, including sources rated the same (one response had to be rejected because it was based on general experience rather than
one project). Again the two responses from the same multi-site collaborative project were treated separately since they were considered to be operating largely autonomously. The number of sources ranked varied from 1 to 17 and when number of sources exploited per project were plotted against number of projects, the number of sources listed by respondents seemed to fit a slightly skewed normal distribution which justified subsequent t tests on the data.

An additional part of this stage of the questionnaire required that respondents provide some detail as to how they went about exploiting each of the information sources they included in their list. Their descriptions are summarised at the end of this section.

a) Scientific/Psychological References on Human Behaviour etc  
b) Psychological or HCI Task Analyses of Related Activities  
c) Surveys/Reports on Human Characteristics  
d) Documentation on Related Activities (e.g. Teaching material or manuals)  
e) Surveys/Reports on Target User-Groups  
f) Specifications of To-Be-Supported Activity  
g) Interviews with Non-Prospective-Users about User Group Characteristics  
h) Interviews with Prospective-Users about their Characteristics  
i) Verbal Task Descriptions from Current Activity Performers  
j) Verbal Task Descriptions from Other Persons  
k) Observation of Prospective User Activity  
l) Observation of Non-Prospective-User Activity  
m) Observation of Activity Independent of System Prototype Use  
n) Observation of Activity Using Prototype  
o) Experimentation with/Testing of Prospective-Users  
p) Experimentation with/Testing of Non-Prospective-Users  
q) Experimentation on Activity with Prototype  
r) Experimentation on Activity without Prototype (e.g. with mock-up)

The list of possible information sources was based upon HCI recommendations for user-oriented design and evaluation techniques taken from the literature in general (e.g. Gould & Lewis, 1985; Jorgensen, 1989), and from the author's experience of Task Analysis, and Evaluative Techniques in practice.
Table 4.15
Incidence of Information Sources Exploited by Respondents

<table>
<thead>
<tr>
<th>Item in List</th>
<th>Incidence</th>
<th>Assigned Frequency Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>n) Observation of Activity Using Prototype</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>k) Observation of Prospective User Activity</td>
<td>14</td>
<td>2</td>
</tr>
<tr>
<td>f) Specifications of To-Be-Supported Activity</td>
<td>13</td>
<td>3.5</td>
</tr>
<tr>
<td>o) Experimentation with/Testing of Prospective-U’s</td>
<td>13</td>
<td>3.5</td>
</tr>
<tr>
<td>d) Documentation on Related Activities</td>
<td>11</td>
<td>5</td>
</tr>
<tr>
<td>i) Verbal T-Ds from Current Activity Performers</td>
<td>10</td>
<td>6.5</td>
</tr>
<tr>
<td>h) Interviews with Prospective-U’s</td>
<td>10</td>
<td>6.5</td>
</tr>
<tr>
<td>a) Scientific/Psychological Ref’s on Human Beh etc</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>j) Verbal Task Descriptions from Other Persons</td>
<td>7</td>
<td>9.5</td>
</tr>
<tr>
<td>q) Exp’n on Activity with Prototype</td>
<td>7</td>
<td>9.5</td>
</tr>
<tr>
<td>m) Observation of Activity Not Using Prototype</td>
<td>6</td>
<td>11.5</td>
</tr>
<tr>
<td>p) Exp’n With/Testing of Non-Prospective-U’s</td>
<td>6</td>
<td>11.5</td>
</tr>
<tr>
<td>g) Interviews with Non-Prospective-U’s about U-Group</td>
<td>5</td>
<td>13.5</td>
</tr>
<tr>
<td>l) Observation of Non-Prospective-U Activity</td>
<td>5</td>
<td>13.5</td>
</tr>
<tr>
<td>b) HCI Task Analyses of Related Activities</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>c) Surveys/Reports on Human Characteristics</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>e) Surveys/Reports on Target User-Groups</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>r) Experimentation on Activity Not Using Prototype</td>
<td>3</td>
<td>18</td>
</tr>
</tbody>
</table>

*U = User  T-D = Task Description  
Exp’n = Experimentation*

A number of possible hypotheses were explored; firstly that university based design teams, R & D groups and HCI consultancy projects might exploit more information sources of the above type than software houses and consultants. Secondly that teams with HCI specialists might exploit more user-oriented information sources than those without. Also thirdly, that better resourced or larger projects would exploit more such information sources than smaller projects.
With respect to the organisational group comparisons, there were only 4 valid responses from Group 5 which was too small a number to compare statistically with the other groups. A one-tailed t test performed to compare the number of information sources exploited by Group A (average number of information sources was 7) and Group B (average 5.44) was not significant at the 5% level ($t = 0.994$ with 19 DF). In other words it would not appear that university based design teams, R & D groups and HCI consultancy projects exploit more user-oriented information sources than do commercial software houses and consultants.

A one-tailed t test compared the 8 design teams returning valid rankings of information sources exploited involving HCI specialists (one other based responses on general design practice experience) against those without in order to determine whether there was any difference between groups with and without HCI specialists in terms of exploitation of information sources. The hypothesis was that teams with HCI specialists would be more likely to exploit more user oriented information sources. Respondents from group C were not included in the study because none of the valid responses came from a team with an HCI specialist, and it was considered that their data, coming from a different group, might bias the result one way or the other.

Groups A and B both contained responses from teams with HCI specialists. In total 8 of the groups providing valid responses involved a specialist and 14 did not. The average number of information sources from the list exploited by groups with an HCI specialist was 9.00, and for those without, 4.78. A one-tailed t test was significant at the 0.25% level with 20 degrees of freedom ($t = 3.32$). This result indicates, rather unsurprisingly, that design teams involving HCI specialists may be more likely to exploit user-oriented information sources, than others.

Project length (in person months, see table 4.3) was plotted against number of information sources used in each project in order to see if there was any obvious relationship between these two variables. What was most apparent was that for projects under two years, there was no relationship between length of project and number of information sources, with number of sources varying from 2 to 10 independently of project length). However, for projects over two years there was a clear relationship. A regression analysis on the data of the 14 projects over 24 person years long confirmed this. A value of $r = 0.734$ was obtained with a standard error of 0.00078 and 12 degrees of freedom. In effect 73.4 percent of the variation in number of
Table 4.16
Ranked Importance of Information Sources Exploited by Respondents

<table>
<thead>
<tr>
<th>Item in List</th>
<th>Overall Weighting</th>
<th>Assigned Importance Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>f) Specifications of To-Be-Supported Activity</td>
<td>113</td>
<td>1</td>
</tr>
<tr>
<td>k) Observation of Prospective User Activity</td>
<td>108</td>
<td>2</td>
</tr>
<tr>
<td>n) Observation of Activity Using Prototype</td>
<td>107</td>
<td>3</td>
</tr>
<tr>
<td>d) Documentation on Related Activities</td>
<td>86</td>
<td>4</td>
</tr>
<tr>
<td>i) Verbal T-Ds from Current Activity Performers</td>
<td>84</td>
<td>5</td>
</tr>
<tr>
<td>h) Interviews with Prospective-U’s</td>
<td>76</td>
<td>6</td>
</tr>
<tr>
<td>o) Experimentation with/Testing of Prospective-U’s</td>
<td>74</td>
<td>7</td>
</tr>
<tr>
<td>a) Scientific/Psychological Ref’s on Human Beh etc</td>
<td>56</td>
<td>8.5</td>
</tr>
<tr>
<td>j) Verbal Task Descriptions from Other Persons</td>
<td>56</td>
<td>8.5</td>
</tr>
<tr>
<td>m) Observation of Activity Not Using Prototype</td>
<td>38</td>
<td>10</td>
</tr>
<tr>
<td>p) Exp’n With/Testing of Non-Prospective-U’s</td>
<td>35</td>
<td>11.5</td>
</tr>
<tr>
<td>q) Exp’n on Activity with Prototype</td>
<td>35</td>
<td>11.5</td>
</tr>
<tr>
<td>e) Surveys/Reports on Target User-Groups</td>
<td>26</td>
<td>13.5</td>
</tr>
<tr>
<td>l) Observation of Non-Prospective-U Activity</td>
<td>26</td>
<td>13.5</td>
</tr>
<tr>
<td>b) HCI Task Analyses of Related Activities</td>
<td>23</td>
<td>15</td>
</tr>
<tr>
<td>g) Interviews with Non-Prospective-U’s about U-Group</td>
<td>22</td>
<td>16</td>
</tr>
<tr>
<td>r) Experimentation on Activity Not Using Prototype</td>
<td>20</td>
<td>17</td>
</tr>
<tr>
<td>c) Surveys/Reports on Human Characteristics</td>
<td>10</td>
<td>18</td>
</tr>
<tr>
<td>( U = User ) ( T-D = Task\ Description )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( Exp’n = Experimentation )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

information sources was explained by project size with a very high probability that there is a relation between increase in project length and number of information sources exploited.

It is difficult to explain the variability in the shorter projects with respect to the number of information sources used. Of the 5 shorter projects with at least 6 sources, only one involved an HCI specialist. There may be some factor which
biases designers towards using more user-oriented information sources which is not captured by this study, or this phenomenon may be mere coincidence. However, one feature of this analysis which should be noted is that the number of information sources used by a design team is not a good indication as to how much time and effort went into exploiting (i.e. selecting, gathering, organising and analysing) information. It may be that in some small teams each individual takes on more responsibilities with each member exploiting user-oriented information however it is very possible that less time is spent on each source than in large teams.

It is possible that both the increased likelihood of HCI specialists’ presence in larger teams and greater resources made available in terms of time and money will be likely to enable teams to make better use of user-oriented information which is available.

From table 4.15 it is clear that Observation of Activity Using Prototype is the most commonly exploited information source in the projects surveyed. Observation of Prospective User Activity and Specifications of To-Be-Supported Activity are the next two most information sources. On the other hand, the most important information source seems to be Specifications of To-Be-Supported Activity followed by Observation of Prospective User Activity and Observation of Activity Using Prototype (see table 4.16).

Experimentation with/Testing of Prospective-Users also seems to be important, but the degree of rigour with which this activity has been pursued is not clear from this study. It seems probable, from the number of respondents who repeatedly used the word "informal" to describe their activities, that experimentation, where it took place was not generally of the type commonly meant by psychologists and HCI specialists. Five respondents did describe what appeared to be more formal experimental evaluative procedures. In three cases HCI specialists were involved, in one of the others the design team was familiar with HCI.

In general, from the findings of part I and part II of the questionnaire, it appears that specifications provided by clients or marketers, documentation from comparable systems, observation of prospective users performing tasks with current systems or on a prototype, and interviews and verbal task descriptions provide the bulk of user-oriented information used in UI design. Experimentation, even of an informal
nature is of secondary importance, and the scientific literature and Task Analysis methods are also secondary. HCI DETs of the type reviewed in Chapter 2 did not appear to be used by any of the respondents.

As with design constraints, respondents were invited to add, and indicate the relative importance of any other information sources which were not on the list.

1A Software porting expertise.
2A Previous product experience.
3A System usability audit.
   Field reactions of users.
5A Conference attendance.
   Other people in company.
8A Suggestions/help from software vendors.
9A Existing products.
10A Technical papers (graphics, windowing software and systems).
11A Hardware specifications.
   Papers on graphics.
   Papers on colour.
2B Personal experience (of researchers/psychologists in organisation).
   Guidelines
   Third hand feedback from potential users (the client's clients).
3B Experience with "spy".
   Experience with previous UI product design.
   Demo programs from a computer & software company that were based on the package used.
5B Existing interactive graphics modelling programs & literature relating thereto.
   Existing demo programs for workstations.
   Computer manufacturers literature and manuals.
7B Our own ideas on how it could be done.
2C Technical support.
   IMS experts.
4C Written descriptions (practices) from users job functions.

These additional information sources were also mostly ranked as important by respondents. They suggest that designers see personal experience, specialist
expertise and existing comparable systems as important information sources which influence their own UI design. However, the study did not capture any information as to how designers were able to determine whether it was appropriate to "borrow" somebody else's design ideas. Designers did not describe how they decided what features were appropriate to copy from existing comparable system, given possible differences between other parts of the new design and the system from which the idea came. It is not clear whether borrowing design ideas causes problems of inconsistency between different parts of a system.

In the following section respondents were asked to elaborate the manner in which they went about exploiting information sources. There were 20 responses to this section in the questionnaire, these are listed in Appendix 2.

In the following, elaborations for each of the information sources listed labeled a to r, and those added by respondents are summarised.

a) Scientific/Psychological References on Human Behaviour etc This source was used in literature searches for information such as human reaction times and colour sensitivity. It seems that such information was useful in fine tuning of designs.

d) Documentation on Related Activities (e.g. Teaching material or manuals). Existing task manuals were used by one respondent to help structure the interactive system tasks. One team used a competitor's documentation to "identify pitfalls, drawbacks, etc from the documented description." Another carried out a survey of similar systems to their design. One respondent stated that this information source was used for developing specifications which were used as design guidelines.

e) Surveys/Reports on Target User-Groups. These were described by one
respondent as being used as initial input. One respondent stated that an advisor to potential users supervisors provided this kind of information. Another found it in the available literature.

f) Specifications of To-Be-Supported Activity. The information required for such specifications was described as being provided by marketing, or obtained by attending a course in the application domain, taken from the professional press, manuals and standards documents, or from investigations of existing tools. Interviews with potential users were also a used to obtain this information. It seems that the design team itself was usually involved if any detailed written specification was produced, rather than being the recipient of a pre-prepared specification.

g) Interviews with Non-Prospective-Users about User Group Characteristics. When non-prospective users were interviewed it seems that they were either application domain experts or, in one case, supervisors of potential users.

h) Interviews with Prospective-Users about their Characteristics. This source of information seems to have provided specific user and task information which helped with detailed aspects of design. For example "...so the operator has the final word on accepting/rejecting shift figures. The system however ensures that all sections have been entered before reporting the discrepancy and sections can be corrected individually, without extensive retyping." Such information might be used for requirements analysis, or for evaluating prototypes. Users seem to be better at evaluating existing designs than at suggesting possible design features; "They were better at reacting than initiating".

i) Verbal Task Descriptions from Current Activity Performers. These were used in much the same way as h.. They seem to be informal but may provide sufficient detail to drive the design; "Talking about the task to be performed led almost automatically to the chosen design. The users had already divided the task into meaningful subtasks."

j) Verbal Task Descriptions from Other Persons. These might be provided by discussions with management, marketing, or others outside the design team.

k) Observation of Prospective User Activity. Prospective users were observed at
work using, in one example, a manual system, and in another, existing tools. In the example where prospective users were observed using existing tools, it was possible to elicit requirements from them based upon the drawbacks of these tools. One respondent without HCI expertise described using unobtrusive observations to "test which types of display were most noticeable/ most readable etc."

1) Observation of Non-Prospective-User Activity. This seems to be much less used as an information source; in the one case where it was described, the individuals observed belonged to the same profession as the prospective users.

m) Observation of Activity Independent of System Prototype Use. As indicated by other descriptions of information sources, observation of use manual systems and existing tools have been given as examples of this source.

n) Observation of Activity Using Prototype. Occasionally this important source of information seems may be obtained by chance in the course of the design cycle. In one case, a respondent stated that potential users ignored the prototype made available for them to try out; "...they ignored the release altogether due to the bugs which were still in it, and the low level of reliability of the tool, and would carry on using existing tools." When it was obtained this information was helpful in driving modifications to designs.

o) Experimentation with/Testing of Prospective-Users. Experimentation, of a scientific nature may have been quite rare, however testing seems to have been quite common. Informal acceptance testing, or more formal detailed tests of comparable screens, response times, seems to have been more prevalent amongst the respondents' projects.

p) Experimentation with/Testing of Non-Prospective-Users. This option was used as well as testing with prospective users on occasions, or when prospective users were unavailable or difficult to find. One respondent described such tests being used simply to resolve arguments over the design when different interest groups refused to give in. There is little evidence to suggest that experimentation was of a scientific nature.

q) Experimentation on Activity with Prototype. In all cases where
experimentation/testing took place, apart from one where mock-ups were used, a system prototype was involved. Sometimes designers experimented with the prototype themselves, then feedback from others, sometimes including prospective users was used for modification. This might take place at various stages in the design cycle. A prototype may be used for demonstrations as well. As above, experimentation seems from the elaborations to be an informal activity in UI evaluation.

r) Experimentation on Activity without Prototype (e.g. with mock-up). Only one respondent was specific about this information source, stating that mock ups were used for very early tests.

The elaborations provided by respondents of their exploitation of information sources tend to confirm the rest of the information from the questionnaire. Once again clients, and collaborators figure as important to the design process in providing many of the information sources listed above. Domain experts or marketers may be the source of specifications of target tasks, descriptions of target users. Prospective users themselves are mainly involved in evaluation, rather than in specification and development. Whatever observations, interviews, experiments and so on, there were appear to be informal in nature.

4.3.15 Subjective Evaluations of Good and Bad Design Features

Finally the questionnaire asked respondents to describe their satisfaction with the outcome of the UI design. The detailed responses are listed in Appendix 3. The rationale for this question was that the responses would be indicative of what designers typically consider to be good and bad features of their UIs. Alternatively respondents could have been asked to list good and bad UI features in general, but idealised responses might have been more prevalent. Table 4.17 summarises good and bad features of UI designs described by designers. Features which are very specific to the details of the project described, such as inability of users to agree on aspects of the UI, are not included. Also concepts such as "Good graphics" and "Good use of sound" are not included because it is impossible to determine what is meant by good.

In table 4.17, no order is implied as to which features are most important. Designers were not asked to rank their responses. However, in Appendix 3,
consistency and simplicity/complexity were mentioned by several respondents. The fact that designers made use of such terms suggests at least some penetration of HCI concepts and principles through to design practice, even if HCI techniques seem to be a rarity.

Speed of response of the UI was also mentioned frequently as a good or bad feature, and appears to be important (probably because slow response times render a UI almost totally unusable, and also because fast responses are often hard to achieve). Some of these features reflect some of the principles of user-oriented design simplicity, compatibility, user centred task dynamics, consistency, observability and retrievability, as categorised in Chapter 1. In fact the only principle which is not represented by comments relating to good and bad design features is retrievability.

Any features which are not related to any of the principles of usability from Chapter 1, tend to be based around pragmatic considerations, such as conformity with in-house style, technological restrictions/compromises, and UI software compatibility with other system software.

4.4 Discussion of Results

Much of the information gathered in this study is of a qualitative nature. As was stated earlier, the study did not attempt to impose preconceived classifications on respondents. The reason for this was that, in essence, this was an information gathering exercise which aimed to discover features of "real world" design practice which might impinge upon the application of user-oriented design and HCI DETs.

4.4.1 Grouping Projects According to Host Organisation

There were few differences between the design projects host organisational groups. The first of those that did emerge was that although no group was seen to include more HCI specialists than others, R & D Groups, universities and HCI consultancies (Group B) design teams seem to be more likely to involve HCI awareness in general than software companies or consultants' (Group A) teams and departments of non-computer companies (Group C). All of the teams in Group B involved individuals with some HCI familiarity, compared with less than half of Group A and only one in Group C. The reason for this is not given by the responses although it seems
Table 4.17
Good and Bad Features of UI Designs
Described by Respondents

<table>
<thead>
<tr>
<th>Good Features</th>
<th>Bad Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniformity/Consistency</td>
<td>Complexity</td>
</tr>
<tr>
<td>Standardisation of Function Keys</td>
<td>Hard to Remember</td>
</tr>
<tr>
<td>Ease of Learning</td>
<td>Exceptions to Rules</td>
</tr>
<tr>
<td>UI Division Matches Task</td>
<td>Inconsistencies between UIs</td>
</tr>
<tr>
<td>Division</td>
<td>Inconsistency between UI modules</td>
</tr>
<tr>
<td>Easy to Follow Procedures</td>
<td>Overloaded Mouse Button</td>
</tr>
<tr>
<td>Minimal Keystrokes</td>
<td>Functionality</td>
</tr>
<tr>
<td>Modelessness</td>
<td>Frequent Mode Swapping</td>
</tr>
<tr>
<td>Window Types/Screen Areas as Modes</td>
<td>VDU Oriented UI Design</td>
</tr>
<tr>
<td>Pop-Up menus</td>
<td>Requiring Computer/Application</td>
</tr>
<tr>
<td>Display of Command Meanings</td>
<td>Expertise</td>
</tr>
<tr>
<td>Dimmed Non-valid Commands</td>
<td>Vague command Descriptions</td>
</tr>
<tr>
<td>Only Valid Options Displayed</td>
<td>Non-Display of Key Press Orders</td>
</tr>
<tr>
<td>Simple Uncluttered Screens</td>
<td>Hidden Required Key Presses</td>
</tr>
<tr>
<td>Fast Displays</td>
<td>Lack of On Line Help</td>
</tr>
<tr>
<td>Robustness</td>
<td>Slow Response Times</td>
</tr>
<tr>
<td>Modifiability</td>
<td>Technological Restrictions/</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Compromises</td>
</tr>
<tr>
<td>Sensible Response to All Input</td>
<td>Technologically Outmoded</td>
</tr>
<tr>
<td>Conformity with House Style</td>
<td>Inflexibility</td>
</tr>
<tr>
<td>Cross Check on Data-Entry</td>
<td>UI Incompatibility with Other</td>
</tr>
<tr>
<td></td>
<td>Software</td>
</tr>
<tr>
<td></td>
<td>Double Entry Cross Checking</td>
</tr>
</tbody>
</table>

intuitively likely that employees who are more frequently engaged in research projects in a research oriented institution or organisation (i.e. those in Group B), are more likely to have the opportunity to discover other fields than their own specialist area.
The second difference is that Group C teams appear to work on less varied applications than the other groups, all of those surveyed being in either data processing or process control. The third difference is that Group A design teams seem to be significantly more prone to technological problems than other groups, although the reasons for this are somewhat obscure.

### 4.4.2 Project Size

Project size varied greatly from 0.67 person months to 1800 person months (150 person years). The three very large projects (60 years and over), two in Group A and one in Group B, meant that average design project size was far greater than the median. The median project length was 33 person months and the median size of design team was 7. These values represent more typical project and design team sizes than do the averages. The presence of a few very large projects made a potential statistical test for the relationship between host organisation group and project size unreliable. However, there were no obvious differences between the groups in terms of project size.

### 4.4.3 Familiarity with HCI

As stated above, design teams in R & D Groups, universities and HCI consultancies (Group B) all involved individuals with HCI familiarity, but they were not more likely to involve HCI specialists. No relationship existed between lay-familiarity with HCI of team members and size of project. However design projects with HCI specialists were significantly larger than those without.

Two of the specialists indicated that they were not able to have as much influence on the design process as they would have liked. This finding is in line with the point raised by Smith and Mosier (1984) to the effect that HCI specialists may lack the knowledge about design and the tools to deal effectively with it. They propose that this may explain the lack of influence which they achieve. It may be that greater influence can only be achieved by those who have sufficient breadth of experience to be able to communicate effectively and convincingly with other members of the design team.

As stated in Chapter 3, smaller, less well resourced design projects will be less able
to apply sophisticated HCI DETs, particularly because they tend to rely on HCI expertise, and a significant amount of time and effort invested (albeit with the promise of rewards in terms of improved results). If smaller projects are less likely to involve specialists, then this problem is even more extreme. Additional, simpler techniques need to be targetted at smaller design projects which appear to be under provided for by HCI methodologies at present.

It would appear from the findings that over 70% of the respondents were at least positive about the potential benefits of HCI. Some of those who were not positive seem to have no knowledge of the discipline at all. The responses suggested that there may be an attitude problem towards the form in which HCI and other specialist discipline’s design recommendations are expressed. Perhaps researchers in the field presenting recommendations and methodologies for use in design practice need to be more aware of the fact that systems designers’ backgrounds and understanding are very different from their own. Much that seems obvious and straightforward to specialists in HCI may be abstruse and confusing to others.

4.4.4 Product Functions and the Type of UI involved

The use of text editors as test environments for HCI DETs (e.g. TAG, GOMS, and CCT) is not vindicated by the results of this study with only one of the applications being comparable to such a system (a word processor). Data processing was the most common application type, followed by process control and office systems. This study tends to suggest therefore that these are the types of application on which HCI DETs should concentrate, since there is no guarantee that success with one application will be generalizable to another very different one, unless the DET is actually tried out on the other application.

4.4.5 The Prospective Users

The great majority of design projects were targeting their UI at users who were quite likely to be computer naive and possibly infrequent users. In other words it would not have been safe to assume that they would share the same view of the UI syntax and semantics as did the designers. This being the case, Gould and Lewis’s (1985) principle of user involvement throughout design appears to be vindicated. The only way designers can hope to be sure of potential users’ understanding and
requirements is to directly involve the users themselves.

Unfortunately the kind of user described (including children, gallery staff, and garage attendants) could not have been relied upon to have been trained to perform tasks in the ideal and error free manner represented by a competence grammar describing their task actions. TAG, GOMS, CCT, and other HCI DETs tend to generate competence grammars or idealised task methods for the sake of economy (acceptance grammars or many, perhaps inefficient alternative task methods are likely to produce exponentially bigger models which will be more difficult to analyse; Payne & Green, 1986). Maclean et al (1985) demonstrated that people are very poor at assessing the efficiency with which they are carrying out interactive tasks. They may believe they are using the quickest methods when, in fact, they are not. This means that in many cases the accuracy of competence models of interaction may not be great enough to justify their application.

4.4.6 Organisations involved

The majority of design projects surveyed involved collaboration with organisations outside the design team's host organisation, or with other departments in the host organisation. Therefore the size of design teams may not be representative of the real numbers of individuals involved in the design process. The constraints ranking indicated that a lack of structure of design approach and roles, rather than too much structure was typically viewed as a problem by respondents. It is possible that the clarification of design requirements and goals, and shared views of the design process itself are difficult to achieve when large numbers of individuals involved. Boehm et al (1984) show that larger design teams are less efficient (in terms of delivered source instructions per person) than smaller teams. The reason for this could well be that larger teams require more communication and organisational activity in order to maintain an integrated approach.

The implication from these findings is that, in some cases, structured analysis and design techniques which account for user-oriented aspects of design could be a welcome improvement. At present HCI DETs do not provide a satisfactory design framework within which they can be applied (see Chapter 3). This inadequacy may well be an important target for future research.
4.4.7 Information from Specifications, Task Descriptions and Users

In total 65% of respondents stated that requirements, specifications were generated for the UI. These were variously generated by the team itself, marketers, clients, and application experts. The majority of these appear to have used informal English, rather than any structured or formal notational device. Furthermore some of these specifications may have been very high level and brief. Overall there was immense variability in the apparent rigour with which specifications had been generated and used. In one case functional requirements specifications were generated as, and after, the code was written, at the other extreme they were generated early and drove the entire design. Likewise information about the to-be-supported tasks for the system was gathered from a variety of sources and typically expressed in an informal manner.

In 32% of the design projects in the study specifications of users' requirements did not come directly from potential users. For example client managers, marketers or documentation may have been the source used by the team for discovering users' requirements. In these cases many simply could not get hold of target users, but in three cases (12% overall) users were simply not involved until late in the design cycle. This is only a minority of the projects studied, but there still appears to be a degree of complacency about user requirements amongst some designers who assume that prospective users' managers (who often come from a different background, in terms of skills and experience, from other company employees) can provide reliable views of user requirements.

The main input from users seems to take place in evaluation of prototypes, or of the final system. Users may not be very effective as requirements generators because of their probable ignorance as to what is and is not possible in terms of system support for their tasks. One designer stated that users were better at reacting than initiating in the design process. However, this is not an excuse for complacency since there did not seem to be any cases where designers actively sought to present users with a view of the functional potential of the prospective system which they could have used in describing their own requirements.

In 9% of the projects no user evaluations took place, and there were no good explanations for this lapse, particularly since the prospective users were likely to be
computer naive.

There was a great degree of variability in the design teams' familiarity with the application domain, with Group C (who might have been expected to be familiar with their own company's application domain) being no better off than the other groups. Designers spent some time familiarising themselves with application areas, watching current activity performers or talking to domain experts. Presumably early design specifications would have to be comprehensible to such individuals if they were to be evaluated for validity, so there may be considerable pressure from this source for informal "English" specifications.

4.4.8 Specifications and Methodologies Used

Forty percent of the respondents claimed that their team had used an abstract design specification or descriptive methodology, some used more than one methodology. Of these, ten responses, only 4 were clearly user-oriented. In general informal approaches were preferred as with the requirements specifications and users/tasks specifications.

It may be that informal specifications and descriptions are preferred by the majority of individuals involved in design. Structured descriptive methodologies and notations, with so many to choose from, require some investment in terms of time and effort to master, until such investment takes place these techniques remain somewhat esoteric and abstruse to the majority. It seems unlikely that all those concerned in a design project will be familiar with the same techniques, so the obvious notational device has to be the most comprehensible by the majority of those concerned. The two quotes to the effect that SADTs are not generally viewed in a favorable light supported this hypothesis. Furthermore, none of the HCI DETs described in Chapter 2, or any related methodology, was used.

This finding is not favorable for HCI DETs since it implies that any methodology which generates specifications which are obscure in derivation and notation is of little value to the design team as a whole. The variety of individuals concerned cannot all be expected to master all applicable methods and notations in case somebody wishes to use them. Presumably "in-house" methods are a partial answer to this problem; at least all the members of the host organisation can use such
specifications.

Another alternative solution to this problem is that the design team is structured in such a way that specialists in various aspects of design have the ability to translate from informal to formal or structured specifications of their own particular discipline and back again. However, the argument against this would be that any descriptive power gained by a formal or structured specification could be lost as soon as it was translated back into an informal one.

### 4.4.9 Generation and Testing

Four keywords or concepts were identified as distinguishing features of the large variety of approaches to generation and testing of the UI. These were *Formal*, used in 36% of responses and which in some cases may have meant structured or rigorous; *Test-against-specs* described to by 20% of respondents, which generally referred to evaluations involving comparisons of performance with prespecified requirements (which may have been informally stated); *Informal*, a term used by 52% of respondents, which referred to unstructured approaches; and *Iteration or Prototyping* used by 56% of respondents.

The general implication is that informal iterative approaches to system development are more common than structured ones. Again, this suggests that there is not any widespread acceptance of the value of rigorous approaches to specification and evaluation of systems in current design practice, and this suggestion applies both to straightforward systems analysis and design techniques as well as to more specialist techniques such as HCI and formal methods. There remains an onus of proof on those who advocate such techniques, to convince designers of the reliable benefits of their application, as compared with methods presently in use.

### 4.4.10 Problems and Modifications

The majority of respondents experienced notable problems in their project. Many of them had to make major modifications as a result. The incidence of technological problems appeared to be significantly greater for Group A (software companies or consultants), although this result is somewhat unexpected and not explained by the survey findings.
The main implications of the probability that a major, unforseen problem is likely to emerge at some stage in the design, are that major modifications may well have to take place, or that there will be unavoidable limitations to the product of the design cycle. HCI DETs do not generally suggest how such modifications may affect their application, for example the user's conceptual model, defined in the early stages of design may have to be violated. There is no guidance for damage limitation in such circumstances.

Technological, organisational, and pragmatic problems are probably not amenable to improvement through the adoption of better design approaches. These represent 13 out of 21 problems listed, or 70% of the total. UI complexity, poor design concepts, and methodological problems, 30% of the total, may well be reduced by user-oriented-design approaches. However, in the majority of design situations, designers will probably have to be prepared for a variety of unavoidable set backs, and applicable methodologies will have to be designed to deal with in such situations.

4.4.11 Finalisation of the Project

System design projects may be finalised by agreement between the parties involved, or by iteration presumably towards a satisfactory system design state. This does not imply that the potential users are really satisfied. None of the respondents stated that the design was completed because it met usability requirements, or because user were satisfied with the outcome. One respondent stated that the project completed when a functional specification was satisfied.

The emphasis may often be on satisfying the client management, or a marketer, rather than users directly. If this is the case, a design team may not be under optimal pressure to ensure that the design product is satisfactory for the potential users. Unfortunately the responses to the questionnaire did not clarify how closely potential users were involved in the finalisation of a product. However if pressure to ensure usability is low, then designers are unlikely to adopt time consuming and expensive measures to improve this feature of a system.
4.4.12 Lessons Learned

Of the variety of responses to the question, what would they have done differently with the benefit of hindsight, one interesting category was identified. These were the 6 respondents who stated that they would have liked to have used a different methodology altogether. Together these responses represent 24% of the total. One of these stated that prototyping would have been desirable, but was not possible at the time of the project. The other five either wanted to improve the influence of user-oriented activities, or improve the methodology of the whole design process, including increasing structure and applying such a methodology earlier.

4.4.13 Constraints on Design Activity

The ranking exercise did not reveal any relation between the number of constraints listed by respondents and host organisation group, project length, and presence of HCI specialists in a design team.

The most common and important design constraint seems to be the complexity of the application or product sophistication (although time shortage was included as an additional major constraint by many respondents and may have been as important). The reason for this may be that design in itself is an intrinsically difficult set of ill defined problems. This may be particularly true of design teams where some of the members are very unfamiliar with the application domain. If the project involves generation of a sophisticated and complex solution, designers may feel constrained by the difficulty of understanding and providing even a single solution to design problems. Where the application is familiar and simple, they may have more freedom to choose between alternative design solutions.

Lack of information about users is also an important constraint as this prevents designers being sure of the validity of their assumptions and decisions. It seems that information about tasks is more readily available than information about users. As we have seen, designers often get their task information from intermediaries, and it is not clear that task information obtained is always valid. The nature of users is of course intrinsic to the manner in which they carry out tasks; novices will make many errors, and require a great deal of help; experts will want high level short cuts for frequent actions, occasional users will have difficulty in building up skill using
the system. These and other features have to be taken into account by UI designers in the structuring of UI task methods.

Over casual evaluation approaches, lack of guidance from others involved in the design, over casual design approaches and undefined team member roles were all constraints of some importance. Such constraints tended to outweigh constraints relating to excessive structuring of methods and roles, and overintervention from others outside the design team. In general this gives the suggestion that design is more typically underorganised and unstructured than the opposite.

The obvious lack of application of structured systems analysis and design methods may be the main problem here. Although such techniques may not directly improve the UI, some of the constraints surrounding design could be ameliorated by their application. Were such techniques to include more user-oriented components, the situation would probably be greatly improved.

Other important constraints include inadequate resources and small team size, lack of assistance from collaborating clients, and lack of experience with design itself and with HCI. All of these contribute to the perceived difficulty of design in itself. Any HCI methods which are also hard to master, can only increase the designer's sense of difficulty.

4.4.14 Information Sources

The ranking of user-oriented information sources for UI design and evaluation exploited by design teams revealed some interesting differences between long and short projects and projects with and without HCI specialists. Projects under two years long showed no relation between the number of information sources exploited and size of project; the reasons for this are not clarified by this study. For these small projects the presence or absence of HCI specialists in the team does not explain the lack of correlation.

However for projects over two person years in length, there is a strong correlation between person-years and number of information sources. This relationship is further strengthened by the fact that projects with HCI specialists use significantly more information sources than those without.
These findings imply that time and expertise are strongly related to application of user-oriented design. This may not be a causal relationship, it may be the case that the perceptions of those involved in the project influence decisions as to whether it is feasible to carry out detailed user requirements analysis, task analysis, and rigorous evaluations, and that when time or expertise is short, they choose to leave out these activities, when there is no real reason why they should.

The most important information source seems to be specifications of to-be-supported-activities. As has been suggested, these may not be reliable if they are not taken directly from discussions with, observations of, or analysis of the behaviour of potential users as sometimes appeared to be the case. The next most important information sources are observations of potential users, and observations of activity with the prototype. These information sources reflect the apparent bias towards unstructured and informal approaches to design and evaluation. The elaborations given of how information was exploited did not suggest that these sources were typically rigorously exploited.

Interviews and verbal task descriptions are also very important as information sources, but again may be supplied by non prospective users. The suggestion is that these too are dealt with in an informal manner. Experimentation with the prototype rarely appears to be of a scientific nature. However experimentation is clearly important as a source of valuable evaluative information see table 4.16. Only 25% of the projects studied seemed to treat experimental evaluation as a well defined, structured activity. (i.e. many respondents described using experimentation, but few described any scientific procedure).

HCI literature and task analysis techniques did not seem to be as important as one might have expected. HCI literature was ranked as the eighth most important source (out of 18). One respondent spoke of the difficulty of obtaining HCI information "While working in industry it was difficult to get easy access to HCI ideas that would be immediately useful...". The implication seems to be supportive of other empirical studies findings to the effect that HCI research and techniques are not generally applicable. As one of the specialists stated "Only pragmatic tools are useful. Data & design guidelines from HCI are almost never of any real use."

Elaborations of the exploitation of information sources by respondents tended to
confirm existing suspicions about various features of design and evaluation. Most notably that varied and informal approaches are adopted, and that indirect information is obtained about users from managers marketers etc.

The additional information sources supplied by designers are mainly comprised of personal experience of design team members, and observations of comparable products. Borrowing of ideas seems to be quite common, although it is not clear that this does not lead to problems of inconsistency, whether it be within the UI itself, or between the UI and any accepted in-house style. At present HCI DETs do not account for this borrowing phenomenon, which designers probably rely on as a means of reducing their own workload.

4.4.15 Subjective Evaluations of Good and Bad Design Features

The results from this part of the study were used to indicate what designers typically consider to be good and bad features of their own UIs. This idea is distinct from that of defining idealised system features because the basis for these judgements is real systems.

Some designers do seem to be intuitively aware of important issues with respect to system usability. Some of the features they identified tend to reflect the usability principles identified in Chapter 1. For example at least five of the respondents good or bad features indicated that they were aware of the importance of consistency. Many of the designers mentioned simplicity. One of the designers referred to user centred task dynamics (UCTDs) stating that the division of the UI matched task divisions.

The intuitive guesses of UI designers may often turn out to be wrong, but this study suggests that many of them have already begun to adopt some user-oriented concepts in their approach, albeit in an informal intuitive manner. This means that designers may already agree with HCI researchers as to what aspects of UI design may be important for users. It seems to be the HCI methods rather than the message which are not getting across to designers in applied practice.
4.5 General Discussion

The preceding discussions have been restricted to the nature of each of the different aspects of the questionnaire analysis. In the following discussion the support obtained for each of the initial hypotheses will be discussed. However some more general implications of the findings can be deduced; these are taken up in Chapter 7.

Validation of Hypotheses

1. The design process is very variable, in terms of activities and organisation.

The findings of this study have supported those of others (Gould & Lewis 1985, Hannigan & Herring 1987) in that the design process has been found to be highly variable in nature. Design projects vary extensively in the kinds of activity undertaken, with few standard approaches being observed. Perhaps the most common feature of design is the use of iterative, prototyping for testing and user evaluations.

Not only do activities vary but applications, target users, team size and roles, and organisational structure are also diverse. Design recommendations or methodologies based upon presuppositions about any of these aspects may prove to be invalid in many cases.

2. Applications are diverse and text editors only make up a small minority of these (this hypothesis queries the generalizability of HCI DETs to a wide range of UIs).

The study demonstrated that design projects vary enormously in terms of the application of the system. The most common application areas seem to be data and process oriented. In this type of system there may be multiple views of the information required by users. Also the behaviour of the system may not be in full control of the user, particularly in real-time, process control systems. Text editors are the most popular candidate applications for testing of HCI DETs, almost to the exclusion of all other types of system, however they are not representative of the majority of design project applications. As has been stated earlier in Chapter 3, text editors are special in that they involve little other than an interface and some text. There is
generally only one view of the data (text) and the behaviour of the system is generally in full control of the single user.

Another point relating to applications is that the most relevant human performance metrics of the type provided by HCI DETs collectively, will vary between application domains. In some domains speed could be more important (e.g. word processing) and in others errors could be more important (e.g. process control). Without considering any other factors, it will still be the case HCI DETs are likely to be relevant to only a subset of possible design projects, purely on the basis of the relevance of the performance metrics they provide.

3. **Prospective user populations are highly variable, and the existence of ideal or even expert users cannot be relied upon.**

It was clear from the responses that prospective users were diverse groups and that many of them were possibly computer naive. The types of users described by respondents, such as gallery staff, garage attendants, and secretaries could not have been relied on to receive special computer training of the kind described by Kieras & Polson (1985). They would therefore be unlikely to be using idealised methods for accomplishing their tasks (Maclean et al., 1985), and competence models of users interactive performance applied as a UI evaluative method could well prove to be highly inaccurate. Furthermore since most users would be computer naive, they would be likely to make frequent errors and performance models such as GOMS and CCT would be inaccurate and could prove inadequate for assessing the causes of errors.

4. **Time and project resources are frequently insufficient for a satisfactory amount of work to be done on the system design itself which affects the UI design also.**

The most common and important design constraints ranked by designers were complexity of application/sophistication of product and (from the additional rankings supplied by designers) limited time. It appears that designers often feel that they have so much to do in a project, with so little time, that it actually constrains their design activity.

This finding tends to belie the fact that only 27% of respondents from completed
projects were dissatisfied with their design product. Another 13.5% were satisfied
given certain qualifications, and 59% were satisfied without any qualification (and
possibly without evidence from proper user evaluations). It may be that designers
are less than eager to suggest that their design was not, in their view a success, or
that they view the constraints which prevent them building better systems as "par
for the course" along with the limitations which this imposes on the outcome of the
design.

The findings do suggest that time is generally considered to be too short, and that
complexity of the application makes it more difficult to achieve good results. How-
ever it appears that in spite of this designers are generally satisfied with their work.
Gaining stronger support for this hypothesis would mean designers having to admit
that their work is unsatisfactory. It is possible that a questionnaire survey is not the
best method for achieving this end.

5. Design teams frequently lack HCI or psychology expertise as would be required
for instance to identify appropriate levels of analysis in a multi level HCI DET.

The proportion of design projects surveyed including HCI specialists was 36% of
the total. A further 20% of teams appeared to have some lay familiarity with HCI,
but not at a sophisticated level. This leaves 44% of the teams with little or no fami-
liarity with HCI at all.

When HCI DETs are evaluated, they are applied by their authors who are highly
expert in their field. The accuracy or correctness of these techniques may depend
very greatly upon the skill of the analyst using them. Sharratt (1987) demonstrated
that CLG was very difficult to use and distracted the HCI MSc students attempting
to apply it from the design itself.

It seems unlikely that most design teams will have the expertise required to achieve
the kind of standards of accuracy reported by the authors of HCI DETs. Further
research is required to determine how accurate such techniques are in the hands of
more typical design teams, and in what way they need to be improved to compen-
sate for lack of psychological and HCI expertise. Sharratt, for example suggests
that automated checking for consistency, mapping between levels in CLG
specifications, and incremental addition or alteration capabilities could be added to
CLG to improve its applicability for its users.

6. Abstract design specifications of any type (including systems analysis and design methods as well as HCI methods) are not commonly used.

The study clearly showed that abstract design specifications, particularly user-oriented ones, are not generally popular amongst design teams. Informal unstructured approaches are more common, and there seems to be a negative attitude towards such specifications; they were referred to as luxuries and unrealistic. Of the 40% where such techniques were used only a small minority were clearly user-oriented, and these did not appear to be of a type similar to the HCI DETs described in Chapter 2. Where abstract specifications are used it seems that they are generally used for modelling states, data flow or for formal verification, rather than task structures, or user representations.

7. Prototyping is a very common feature of design projects.

Prototyping was the most popular method of generating, testing and user evaluating systems. Fourteen of the projects described (56% of the total) involved prototyping and iteration as opposed to 9 (36%) projects involving formal structured techniques, and 5 (20%) involving more rigorous tests against specifications.

8. Designers are not rigorous in ensuring user-oriented design and tend to favour casual and late evaluation, rather than user-driven design and early evaluation.

Although most of the projects studied (all but three) involved user evaluations, most of these were informal, with some doubt as to the degree of rigour with which these were conducted. The purpose of user evaluations was sometimes purely to make the best of a bad design concept, or resolve disputes between interested parties, and in one case a respondent stated that there were no clear objectives in the prototype user tests. In at least 7 cases (28%) user evaluations were delayed until late in the design cycle. However in 6 cases (24%) early evaluations did seem to take place.

Overall this hypothesis does receive some support in the sense that user evaluations tend to be informal, and medium to late user evaluation is far more popular than early evaluation. The value of early evaluation was illustrated by one respondent
who stated that the first exploratory usability test, which took place after a considerable amount of design based on a weak initial concept, convinced the design team that the system had to be greatly modified. The result was that a lot more usability testing was necessary than if the initial design had been guided by user-oriented techniques.

9. There are many constraints which could pressurise design teams towards certain methods and away from novel unproven methods.

As has been stated designers in this study found that, amongst many other constraints, the complexity of the application or product sophistication, inadequate resources and time were the most important. Although respondents did not explicitly state that they were prevented from exploring unproven methods, they did indicate that these constraints limited what they were able to do. It would appear that since many of the novel HCI DETs require considerable time and effort, particularly for complicated UIs, they are unlikely to be adopted by designers who already have problems of this type.

Two of the HCI specialists found that organisational structures (i.e. their position in the company, and the ability of others to ignore their recommendations) significantly reduced their influence over the design. In these cases and presumably others, the attitudes of certain team members could have prevented novel HCI methods being used.

An additional point which is worth making here is that designers see personal experience, specialist expertise and existing comparable systems as important information sources which influence their own UI design. It may well be that, rather than using a structured design technique, or some user-oriented design or evaluative technique in order to enhance the usability of a system (which seems to be rare), design teams prefer to employ HCI specialists (who themselves do not necessarily apply HCI DETs), or use their own experience of previous or competing designs' solutions to deal with problems relating to usability.
4.6 Conclusions

The nine hypotheses regarding applied and commercial design practice, as it relates to the UI and usability were largely substantiated by the findings. Some of the most interesting findings include the following: Design projects appear to vary along a large number of dimensions. Design teams typically operate under a number of constraints which may prevent them from using the most ideal appropriate methods to achieve their aims, which includes user-oriented techniques, as well as other methods such as requirements specifications techniques, functional specification methods, formal verification and so on.

HCI DETs, such as those described in chapter 2, do not appear to be used by system designers in general, and particularly by those without HCI specialist knowledge. However, a variety of informal methods are used for exploiting information sources. Finally psychological and HCI specialist skills are not typical amongst system designers.

This analysis has not aimed to identify the nature of user-oriented design as a process. It has restricted its aims to identification of various activities, and difficulties experienced by designers attempting to design a usable system. In the following chapter a supplementary, more qualitative analysis is presented which seeks to reveal some examples of design as a process with a history, in which activities and constraints tend to be causally related to later problems.
Chapter 5

An Interview-Based Investigation of Applied and Commercial Design Practice Activities and Problems

5.1 Introduction and Rationale

In this chapter a more contextualised study of user-oriented design practice based upon interviews with designers is reported. The features analysis reported in the previous chapter, being based upon questionnaires, did not clearly suggest the context of the projects involved. The work reported here therefore represents a smaller supplementary study which fleshes out some of the features of design practice reported previously. It provides a more broad and integrated view of design as a process where early decisions and activities influence later outcomes. It also focuses more closely on the nature of and reasons for the activities, main constraints and problems encountered by designers which might obstruct the application of good design principles and, more specifically, whether commercial UI design reflects HCI DETs' design views.

The discussions in section 5.4 relate the findings of the interviews directly to some of those in the features analysis in order to give a more general picture of applied and commercial user-oriented design practice as a whole.

The main goals of the study can be summed up as follows:

* To present a more integrated view of applied, commercial practice than obtained by the features analysis of design projects.
* To determine whether HCI DETs or principles are applied by non HCI specialists in commercial practice.
* To explain the design constraints and their effects which might prevent systems designers from applying more user-oriented design techniques.

The targets of the study were applied, and especially commercial design projects. The features analysis indicated wide variations between design projects along a
number of dimensions, such as application type, prospective user groups, host organisation and so on. The selection of target design projects was determined by their distinctness from one another with respect to the system application domain and potential users, the host organisation employing the designer, and the size of the design team and project. Unfortunately it was difficult to find designers who had the time and enthusiasm to take part in the study. Designers who agreed to take part may have been somewhat unrepresentative in terms of their sympathy for HCI research. However, by attempting to select projects on the basis of their diversity, it was hoped that the findings would not be related to a particular kind of design project and would be more representative of the variability of design projects in general. The interviews focused specifically on designers attitudes to, and approaches to UI design, also on the problems they face, and the difficulties which affect their success in dealing with them.

Some of the literature on design practice seems to suggest that systems designers' attitudes and approaches to UI design are commonly negative (e.g. Dagwell & Weber 1983, Hammond et al 1983, Gould & Lewis 1985). Designers are seen as giving inadequate consideration to user issues, and they do not adopt recommended approaches which would increase the likelihood of producing an acceptable interface. The features analysis of UI design described in the previous chapter suggests that it may be more fair to suggest that designers attitudes to HCI are really quite positive. However the main problem, as suggested by Smith & Mosier (1984) appears to be with the knowledge of systems design held by specialists in HCI, some of whom may work as consultants, and some of whom produce recommendations and methodologies for UI design to be applied by systems designers. HCI specialists and their interests may presently represent a minority view in the design process as a whole. As a minority view, it may be up to them to come to a better understanding of the design process and adapt their methods to suit it, rather than to expect the design process to be changed to suit their methods.

The features analysis provided information about what kinds of activities take place in applied and commercial design practice. However the nature of data collection, i.e. a questionnaire survey, was not well suited for pursuing sequences of events and organisation of the design process in any great detail. A better picture of organisation will suggest more clearly the probable scenarios within which HCI knowledge might be applied.
The features analysis also indicated that design teams typically operate under a number of constraints which prevent them from taking an ideal approach, and from carrying out all of the activities which they might wish to. On the other hand, the influence of design constraints on the problems which emerge in design, and on exploitation of potential information sources which might help to improve the usability of the UI, were not made explicit.

It is suggested here that there may be significant and unavoidable problems for the design of usable system interfaces, even for designers who are supportive of HCI and want to tackle user issues. As Gould and Lewis (1985) suggest, the process of system development is unpredictable; there are usually unexpected factors involved which may make it difficult to adopt certain approaches or follow any principles, (the principles of Gould and Lewis are specified with this fact in mind).

The features analysis showed that user-oriented and HCI approaches do not appear to be used by systems designers in general, and particularly by those without HCI specialist knowledge. On the other hand generally positive attitudes were detected towards the aims of the discipline. This could imply that there are other factors which prevent systems designers in general from adopting more scientific user oriented methods, since their attitudes suggest that they can see the value of the discipline, if not its methods. It may be that any constraints which operate on design represent important factors in preventing the uptake of scientific HCI recommendations and methods. By scientific I mean anything which is based upon theoretical and empirical foundations, as opposed to intuitions and heuristics.

In this study HCI specialists were not interviewed as it was considered that their familiarity with and any use they might make of HCI design and evaluative methods would be representative of their own skills and interests, rather than of the pressures of the design process in general. Also, as has been stated previously in Chapter 3, one of the problems with HCI DETs is that they require considerable specialist expertise from those who apply them, and as indicated by the features analysis of design, such skills are not typical amongst UI designers.

As with the features analysis, each interviewee was restricted to discussion of one design project only, since a general discussion would not have been representative of a coherent, integrated design process. Following a number of casual pilot discus-
sions with academics at Queen Mary College with applied and commercial design experience, four formal interviews were conducted with designers who were then working in academic institutions. These involved three employees of London University, and one Brighton College of Further Education employee. All of these interviewees were, at the time of the study, working at Queen Mary College, and were therefore available for further consultation when necessary. Three of the systems designers from the academic institutions were familiar with HCI. However they did not have any formal training in the subject, nor did they have much experience with cognitive psychology which is very important to the understanding HCI DETs (see chapters 2 and 3).

Three of these designers based their interviews on previous commercial system design projects (i.e. the designer was paid to write the software for a commercial organisation, either for its own use or to be marketed) during which they worked as consultants to other organisations. However one referred to a non-commercial, but very much applied (i.e. not research), project. These academic based designers’ interviews covered the following projects:

1. A display editor running on a unix system; the designer developed this editor, with advice and assistance from other members of his university department including computer scientists, and feedback from potential users with a wide range of computer experience also within the designer’s department. This was the only non-commercial project included here.

2. A network management system, for multiple users employed specifically to monitor and control the information being passed through a network. The designer worked largely alone on this project, receiving relatively formal instructions and information about the nature of the task, which did not yet exist in any form, and user requirements from the management of the client group.

3. A garment pattern graphical design-aid or CAD system for fashion designers with limited experience of computer technology. This designer devoted a great deal of time to learning about the task of garment design because of its complexity and unfamiliarity.

4. An educational graphics system for children, aged 5 to 15, to design a figure
which could dance to music. The designer worked alone using specifications provided by the marketing client. In the event the designer deviated from the specifications to generate more appropriate ones.

As indicated by the features analysis, the host organisation may be related to the type of application being designed, and design team size may affect the approach taken towards UI design. The academic based designers were usually working alone on a consultancy basis, rather than as members of a design team. To ensure that representative information was collected, four individuals in commercial organisations were also interviewed. Since these people were not easy to contact for further information, a more exhaustive interview procedure was adopted, based on the structure used by Hammond et al (1983) in interviews with commercial designers. These interviews were taped for later analysis. They include the following:

5. A window manager to be used by programmers, produced by a team of a systems architect, a software designer, and a consultant with HCI experience. The window manager was designed for a Unix workstation with a graphics display. The HCI consultant provided a catalogue of interactive techniques recommended for good interfaces. The other two team members did not have a great deal of HCI experience themselves.

6. A systems designer working for a company which sells products on a wholesale basis. The aim was to computerise the records of orders and sales and provide a word-processor for report and letter writing, all presented as an integrated office system. This designer used an applications generator on a less advanced PC system, but had extremely easy access to the prospective user population.

7. A simulation training device for process controllers produced by a large hierarchical organisation. The designer delegated software writing to programmers and worked largely alone on the actual design, with a manager providing requirements specifications, and a subject matter expert informing him on the details of the application which was being simulated. This designer was required to produce many design specification documents which were verified by the manager and by the client for whom the product was being designed, be-
fore the programmable-ready material could be written as software.

8. A distributed building management system sold to monitor and regulate temperature, locks, lifts etc, to be operated by a very wide range of users, from night watchmen to programmers. The design team consisted of the group leader who was interviewed, four software engineers and a hardware engineer, thus comprising the largest design team studied.

5.2 Interview Structure and Methodology

A similar philosophy was adopted to that of Rosson et al (1987), in that the main aim of the study was to gain as much information as possible, rather than to confirm preconceived hypotheses. For this reason the interview structure was flexible without the content being permitted to become too general. The interview was carried out in the normal work place of the designer, and in all cases the interviewer was able to inspect the result of the commercial design project (i.e., see a demonstration of the UI itself). The interview was conducted in a series of stages which were structured as follows:

* **Stage 1.**
  * Designer describes general tasks supported by the system
  * Designer describes envisaged user population.
  * Designer describes own role in design process with respect to impact on UI.

Interviewer relies on checklist of points and questions to ensure appropriate coverage. Categories dealt with were; *user population, applications, system information presentation, input devices, input methods*, and *user's system model*. The checklist was used to increase detailed information and ensure consistency in the areas covered in interviews.

* **Stage 2.**
  General and specific points concerning design decisions discussed. Interviewer includes particular interface characteristics of:
  * The primary system
  * Sources of information used in design
* Constraints on design activity

Designer determines content. Interviewer uses checklist of general and specific points relating to the UI under discussion. The UI itself is referred to, during the interview, for clarification.

Stage 3.
Designer discusses design philosophy and issues more generally.

The interviews with the QMC designers were recorded in note form on paper. The interviews with the designers in commercial organisations were taped as it would not be easy to talk to them subsequently. After gathering the information the similarities between the projects they described, the activities they carried out, and the problems they experienced were identified.

5.3 Findings

The emphasis in the interviews was always restricted to the design of the UI. However in all of the projects the UI was produced within an overall system design such that there was no clear distinction between the UI and the application functionality. Some of what was discussed included the scoping of software functionality (i.e. what the system software would support), and not just the UI to this functionality, so the distinction between UI design and general system design was not a clear cut one. This reflected the roles of the interviewees who were all systems designers with additional responsibilities as UI designers and evaluators.

A major point that needs to be made here is that it is clear that systems designers have to consider all of the evaluation factors described in Chapter 1. They are responsible for the effect that their system has on users. They have to consider how the application behaves and how this relates to the UI. They design the UI to mediate between the user and the system, and are responsible, if not always perfectly so, for the ability of the system to support target tasks to an acceptable level of performance. Interviewees talked about all of these factors, and there was no evidence to support a hypothesis that they concentrated only on one aspect.

The designers interviewed provided much qualitative information about the nature
of design. Some of the problems and the comments they made were strikingly similar. However in the main each design project was quite unique. Only one project involved use of a publicised structured methodology. All of the others used idiosyncratic approaches (either personal, or in-house), which meant that no two projects shared a common structure.

The academic based designers worked largely alone as consultants, which might explain the idiosyncracy of their methods. Also as a result of this they did not experience problems associated with being a member of a team. However, in line with the findings of the features analysis of design projects, these individuals generally had greater awareness of, and knowledge about HCI as a discipline, this was due to the fact that they had the opportunity to work with and communicate with HCI specialists in the academic environment. However only one of the eight designers interviewed could have been described as very experienced in HCI.

5.3.1 Eight Design Scenarios

The interviews provide eight short design process scenarios. These scenarios are not meant to highlight the nature of development of the systems and their functionality. They illustrate the more organisational characteristics of design, the intent of the designer(s) and the logical progression from an initial undertaking to a completed product. A more detailed view of the development of the UI, concentrating on the functionality, user operations and dialogue could demonstrate that certain HCI DETs would be appropriate in theory, however it is practice that is the main interest here. At this higher level of generality it is possible to determine why perhaps certain HCI methods would have been appropriate or inappropriate in the practical situations in which designers found themselves.

Project 1
Display Editor for Academics and Office Staff
in a Computer Science Department

The designer worked in a computer science department in a university and, whilst doing a great deal of document preparation, saw inadequacies in existing text editing packages, and that the system supporting the present package was capable of supporting a better one. It occurred to him that he should write
a better system himself. With his own ideas, ideas from other products, and suggestions from colleagues, he saw the opportunity to produce an improved text editor. He was familiar with some of the field of HCI, but had few resources for this project.

The system functionality was designed to be better suited to editing tasks, and users' requirements. The overall design process was more or less top-down in nature, however it was not formalised. The designer maintained a number of principles which he believed would ensure that the system was easier to use, including WYSIWYG, typewriter metaphor, modelessness, simplicity, and adequate feedback. However these ideas had to be heavily constrained by the many limitations of the system hardware. These limitations had a considerable impact on the nature of the UI, often making ideal solutions impossible.

The prospective users of the text editor would be academics, students and departmental administrative and secretarial staff, representing a wide range of computer familiarity. For this reason the designer aimed to ensure that the editor would be usable with very little knowledge, but that experts would be able to discover more powerful sophisticated commands as they went along.

Since these users worked in the same building as the designer, it was possible to allow them to use very early versions of the developing system. The design process was not structured, and the designer worked alone on the design, generation of the software and documentation for the text editor, but the prospective users (including the designer) were able to meet with the designer and discuss problems they had and further requirements. The design was informally iterated and continued to undergo modifications until the designer no longer wished to continue with the effort. Within its limitations the resulting system satisfied both the designer and its other users.

Project 2
A Network Management System
for Multiple Network Management Staff

The designer worked as a commercial software consultant and was employed to design a network management system for a commercial company who had
developed a network system for themselves. In initial meetings the designer was given descriptions about the network managers' (system users') tasks by a company manager who was very familiar with the existing network system. No representative prospective users were involved in these meetings, and the designer was suspicious that the users would have different requirements from those specified by the manager. He was highly experienced in HCI, but not in psychology, his awareness of user issues may have enabled him to prepare better for the eventual problems which emerged.

The UI to the network management system was to involve a query language for interrogating a database, and a dynamic display representing the network system. The initial questions for the query language were decided by the company sales team and the technical team who had been building the network system. The designer had further discussions with a salesman and the company paymaster, to determine what queries would be required. This activity was carried out using a mockup of the UI. The dynamic display was determined by what the designers and builders of the existing network system thought the network managers would need to know.

The designer built the network management system in such a way that the UI was modifiable. He maintained a rigid structure for the queries in order that users would be able to predict dialogue structures from a subset of the language. The dynamic display was kept fairly rigid because the designer was more confident about the requirements for the display.

Only after the prototype was built and demonstrated to the network managers themselves were they able to say what they needed to see on the display. The network managers then stated that they wanted to be able to ask many other questions of the database. Luckily the flexibility of the UI was sufficient to permit many modifications to be made to satisfy the users. However there was no evaluation of user performance with the new system. The designer's client only carried out a market evaluation of their new product.
A commercial company employed the designer to write a CAD system for them. In the initial meetings the designer met the company director to discuss the nature of the required system. Unfortunately the the director was not experienced in the application domain (originally pattern grading), nor was he experienced with the type of functionality a CAD system might be able to provide. This led to communication problems with the designer which were compounded by the fact that the designer was very unfamiliar with the application domain.

The designer had to read about, and educate himself on the application domain before going on to develop a prototype. Part of this education process involved watching pattern designers at work. After an initial prototype had been generated, the director of the client company saw what was possible with a CAD system and expanded the requirements. Prospective users seeing the prototype also suggested further possible requirements. The designer saw the main problem as being the fact that nobody knew what the tasks involved in interactive pattern grading would be like, this meant that requirements were often driven by development. Potential users were so poor at communicating their requirements that the designer had to go to people working in industrial pattern design for more concrete descriptions of the tasks the system would have to support.

As a consequence of this, the conceptual specification which the designer had of the system evolved along with the prototype. User requirements continuously drifted as the prototype acquired more functionality, and users began to use it for a wider variety of tasks than originally envisaged which the designer also tried to support. He always tried to maintain what he described as "well formed metaphors" in the representation and behaviour of the UI which naive users would find easy to capitalise on. Eventually code that had been written early in the development cycle became an obstacle to further modifications, and the designer had to settle for partial solutions to requirements. He began to wish that he could go back to the beginning and start again.
The designer always regarded the developing system as a disposable prototype, whereas the client company saw it as a product. The client, responding to market pressures, forced the completion of development and ‘improved and packaged’ the prototype. The designer was not happy with the state of the finalised system. He also noted that, in ‘improving’ the final product, the client company made the UI worse by merging two menus which represented sets of operations related to two different types of task.

This designer stated that he was aware of some of the HCI DETs of the type described in Chapter 2, but was convinced that they would be too time consuming to use in practice, particularly CLG with its multi-level respecification approach.

Project 4
An Educational Graphics System
for Children Aged 5 to 15

The designer was employed as a freelance programmer by a software marketing company to code a specification which they themselves had prepared for an educational interactive graphics package. The specification of the target users was fairly straightforward; they would be home computer experienced children with a fair degree of familiarity with the conventions of the type of system on which the package would run. The designer was given the freedom to adapt the specification where required.

The designer attempted to adhere to well formed metaphors which a child might capitalise on in learning to use the system, and to ensure that any modes which occurred simply reflected the functionality and tasks of the users. The package was specified by the client using very informal diagrams and descriptions. However the designer quickly discovered that the specification was unworkable, and had to modify it repeatedly. He considered the worst case, most naive target user (a 5 year old naive child) and attempted to ensure that the system could be used by this user. Without any access to prospective users he had to rely largely on intuition, previous experience with solutions to similar problems, and other products.
Since the marketing company's specification was so poor, the developing system diverged considerably from the original plan. The designer found it necessary to argue every point with the client company, but some of the time the designers' decisions were overruled and, with these disagreements and the inadequacy of the specification, the project overran. The designer did not see the system evaluated for usability, and any feedback about its success came from the software marketer; the designer's client.

**Project 5**

**A Window Manager**

**for Applications Programmers and Graphics Workstation Users**

This project was undertaken by a team of two designers working in a small software and hardware company, with the assistance and guidance of a consultant with considerable expertise in HCI and UI design. One of the designers conceived and developed the original system structure which was then discussed amongst the team. The consultant then wrote a document containing interactive methods for the UI of the application (a window manager for a Unix and graphics workstation).

The other designer used these software and UI specifications as a basis for the development. The UI was designed using low level components which were modular and therefore would be easily modifiable by experienced system users who would probably be writing their own applications for the system. This designer used the experience and examples provided by the consultant to guide design decisions. However no user testing was carried out. Instead the designer had to ask himself "How would I feel if I was a relatively inexperienced person presented with it [the UI]?". The system was informally prototyped and tested for robustness by the two designers in the team.

In spite of the input from the consultant, with checklists of user considerations, the final product when implemented was unsatisfactory. Many users had difficulties with it. The designer noted that he had become aware that there were a number of major problems with the UI including overloading of mouse button functionality, and limitations to the ways in which windows could be manipulated. He stated "Ideally we should be employing someone ... who
understands these issues and knows how to analyse them”.

Project 6

An Integrated Stock Control Database and Word Processing System for Staff in a Wholesale Merchandise Company

The designer in this project worked for a wholesale marketing company and had been employed to build an integrated office system for his organisation which would handle a database for recording stocks, orders and sales, together with word processing for report and letter writing. The specification for the requirements was based upon current methods for doing the stock control and administrative tasks, and on the business requirements of the company. The hardware and software with which the designer had to work had already been purchased by the company.

Since the designer was working in the same office as the potential users, they were closely involved in the requirements specification. The designer used some of the JSD recommended techniques to guide the initial analysis of the functional requirements of the system and a pre-designed 4GL to implement it. The 4GL, the software and the hardware selected by the company all imposed heavy limitations upon what was possible for the designer.

Once the core features of the system had been implemented, as specified within the JSD approach (Jackson 1983), the designer broke from the structure of the JSD methodology and engaged in iterative prototyping which involved getting users to repeatedly, informally evaluate the system. The designer tried to address any suggestions or complaints they made, and also to ensure that the system was as easy to use as possible. Unfortunately the software he inherited was already inconsistent (for example the operation of item selection from menus varied). A further major problem was that the system would not permit word processing to be interrupted in order to access the database and carry out other tasks before returning to word processing. This was a major drawback because the office tasks carried out by users were necessarily varied and interrupted; for example a user editing a letter would be expected to answer the phone which might then mean that he or she had to access the database to answer queries about stocks, or input new data.
The system was not completed at the time of the interview. It was to be used only within the company, which meant that it could be constantly modified and improved by the designer after implementation.

Project 7
A Simulation Training Device
for Trainee Process Controllers

The designer worked full time within a specialist simulation software company on a simulation training device which was being developed for a client of the company; the client was involved in a high risk process industry. A detailed functional specification and a requirements document for the computer based training device was provided by a marketer. The designer was supervised by a manager who checked his work, and supported by software programmers. He also worked with a subject matter expert who explained the nature of the application and users' tasks to be simulated.

No well known design methodology was used on the project. An analysis and design document specifying general aims, user population, resources and time-scales, was used as a reference during the project. Throughout the design process the subject matter expert was consulted and the designer found that his own ignorance of the application domain was useful because it forced the expert to be precise in his descriptions. The designer used specification and requirements documents to produce an early functional system specification which was signed by the client. Then a more detailed specification was prepared which represented the states and transitions of the system and the appearance of each of approximately 500 screens. This specification represented programmable ready material which could be passed on to the programmers.

Prospective users were brought in to repeatedly evaluate the product whenever possible, this was made possible by the close collaboration with the customer who had access to such users (employees). The prototype trials were well structured with pre-tests to ascertain users' technical knowledge, followed by test modules prepared by the client. After the tests, users were then asked to answer some questions about their experiences with the prototype. However the designer still considered that more user feedback would have been useful.
He felt that prospective users were better as evaluators because management perhaps know too much about the application domain.

Iterative user evaluations were continued through to the finalisation of the product when the client was satisfied. Since the designer was not very experienced, and the product itself was the first of its kind, deadlines were left more flexible than would have been the norm.

Project 8
A Distributed Building Management System
for Security, Maintenance and Engineering Staff

The designer was a group leader of a team of one hardware and 4 software engineers (making this the largest team in the study) in a company which specialises in the development and marketing of distributed building management systems to be used by various maintenance and security staff in a building. It was decided by company management that the current product design needed to be updated, however the way in which this should be done was left to the design team.

The designer attended meetings with representatives from around the company in order to collect requirements for the new version. He also analysed the facilities available on the existing system which were then fed into the functional specification for the updated version. Unfortunately users’ requirements were unobtainable because they were not the buyers of the product and therefore had no contact with the company marketers.

The designer produced a more detailed specification from the early specifications. This consisted of an 8 page requirements specification and a 20 page console specification which was based on a windows applications support library marketed by another company. This specification was then verified by company management.

The main requirement for the UI was that routine interactions, such as switching boilers on and off, should be ‘glove easy’ and should not require any learning so that security staff could easily carry out routine tasks with it. The
system also had to support more sophisticated maintenance and analysis interactions with engineers and plant managers. The system was broken down into functional blocks during development and each block was tested separately before integration into the final product.

Unfortunately there was no direct contact with any of the prospective users throughout the design process. Feedback on the development came from people who had contact with users of the existing system, and from in-house demonstrations. The manuals for the system were produced by another company. The designer stated that ultimately low cost and fast system response times were the main design goals rather than low user error rates.

5.3.2 Projects Summary

Overall, the designers based in academia were generally more familiar with HCI than those in commercial organisations. Approaches to design were largely informal (perhaps a methodology was followed to some extent, or some principles used, but the general picture is not a highly structured one). Most notable was that no two projects used the same approach, each one was unique, but some general features of projects were shared. In the following, the main features of interest from each project are summarised.

Project 1
Non-commercial project
Designer had some HCI familiarity
Based design upon explicit usability principles
User input and feedback (informal testing and meetings)
Iterative approach
Very few resources
Designer worked largely alone

Project 2
Designer working as consultant
Designer very familiar with HCI (but not psychology)
Late involvement of users
Used a mockup for early feedback (not from users)
Built in modifiability
Not highly iterative

**Project 3**
Designer working as consultant
Designer had some HCI familiarity
Novel application
Designer unfamiliar users’ with task domain
Highly iterative
Informal user testing of prototype (as early as possible)
Unmodifiable software problems
Forced finalisation

**Project 4**
Designer working as consultant
Very poor requirements and functional specification from client
Designer worked in isolation
Iterative modification of specifications (achieved by fighting)
Project overran
No user involvement or feedback

**Project 5**
Designers employed by company marketing product
HCI consultant provided interactive methods and checklists of user considerations
Informal approach
Highly iterative approach
Modifiable low level components of code
No user evaluations or involvement
Feedback from users indicated some problems with UI

**Project 6**
Designer employed by wholesale marketing company
Supporting current tasks
Very good contact with users
Used JSD and 4GL
Informal user evaluations
Iteration
Heavily constrained by technology

Project 7
Designers employed by specialist simulation software company
Applied in-house design approach
Top-down specification
Specifications signed off by client organisation
Designer worked closely with subject matter expert
Structured, but iterative user evaluations

Project 8
Designers employed by specialist system marketing company
Product was an update on existing release
Detailed functional specification verified by company management
Used "glove easy" principle
Block design (each block tested separately)
Only indirect user feedback from users on company products
No actual user involvement in design

These design scenarios clearly demonstrate the variability between projects and emphasise the difficulty of attempting to present a representative view of design practice in general without ignoring so much information that the whole process is trivialised. In essence this information strongly confirms the suggestions from the features analysis that design practice differs widely between projects. It also indicates that designers vary with respect to the amount of consideration and effort they devote to user-oriented design and evaluation.

Systems designers were seen to be responsible for all areas of design across the system which may explain the diversity, in the extent to which they concentrated on user issues. If they had many other issues to deal with, it is possible that the emphasis they placed upon user-orientedness of their designs might be diminished in their perception of what was most important. However they might not have all responsibilities if they were working in a team. For example they could be under the supervision of a senior designer or manager, or be responsible for the direction of others on the design team (as in projects 5, 7 and 8), in which case there could be
diffusion of responsibility for user-oriented aspects of the system unless, as in project 7, client organisation(s) emphasised this aspect by insisting on some user-oriented design measures such as structured evaluations.

5.3.3 Categories of Design and Development Activity

The main general aim of this study was to provide a more integrated picture of the process of UI design than was furnished by the features analysis described in the previous chapter. It should be recalled that the UI design scenarios evade a possible distinction between UI design and design of the underlying software and its functionality. The interviewees were responsible for both aspects of the system, however the interviews were not sufficiently lengthy to permit extensive discussion of the fine details of the distinction between them, and how much each was responsible for the nature of the other.

In the following discussions the term design, unless otherwise specified, will be used to refer only to design of aspects of the system and its UI which were seen by interviewees to affect its usability. This included such aspects as the scope of the functionality (i.e. scope of the application being designed), and robustness of the software as well as the presentational and interaction aspects of the system. In other words the focus of the discussions is more accurately user-oriented design which includes UI design and some other aspects.

The results of the interviews indicated that design environments and cycles vary along almost every conceivable dimension. Capturing commonalities may be restricted to intent rather than observable activities which are many and diverse, as suggested by the features analysis. Furthermore there were no discrete stages, with the design process being characterised in general by iteration of a variety of activities. In the interviews it was impossible to pin designers down to identification of any discrete stages in the design process. However many of the activities they engaged in showed apparent similarities in their rational intention.

By referring back to the scenarios these rational commonalities become more apparent. Each of the designers went through a process of commitment to solve some obvious problems (as in Project 1) or to a contractual agreement. There was then a stage of self education and/or clarification of the design space or problem area and
the way in which the design might fill the space or solve the problems; this might involve ideas, suggestions, meetings, observations and was in essence a period of information gathering and collation. Then a prototype was generated in all cases which would be iteratively evaluated and improved, until all development activity ceased.

For the sake of simplicity and clarity these rational activities are grouped into five categories of development activity which appear more discrete than is the case in practice. These are labeled as the following: commitment to a design undertaking, conceptual specification, generation of a working prototype, testing, and finalisation. These categories of design and development activity are distinguished by their goal, rather than their observable character. They cannot be described as stages in a well structured and ordered process because they do not occur in a fixed order (and the commitment stage may only be an observable reality in design projects undertaken for external clients) where a paper contract is eventually drawn up and agreed upon. In fact some of them, particularly the conceptual specification, generation of prototype and testing activities, may occur in parallel and the dependency relationships between them may vary somewhat. A brief description of each activity follows.

**Commitment to a Design Undertaking** has contractual importance for projects where there is an external client involved. Clients may have a predefined set of design requirements, however their specificity and how realistic they are, may vary. In project 3 a client director was unclear about the nature of the tasks the system should support, in project 6 a more detailed specification of requirements was available based upon current task methods. The physical activities may involve employment of the designer, meetings, drawing up proposals or contracts, or simply thinking as in projects 1 and 5.

It implies agreement between the designer or design team, and the organisation commissioning the design (which may be the employer of the designer(s)) on an initial set of client-requirement specifications and design-team undertakings. The degree of formality of this activity seems to depend largely on organisational policies. Basically, certain responsibilities of both designer (e.g., keeping to time and cost estimates) and client (e.g., providing support or information) are made more or less explicit. For example, the network management
system designer received a fairly detailed description, from one of the client’s management, of the prospective users’ tasks which the system would be expected to support. This type of activity does not appear to occur to such a formal extent in projects where the design team or designer is producing a product for their host organisation (either to use or market).

**Conceptual Specification** is the process of deciding what the detailed requirements of the system users are, and if necessary any organisational requirements, and deciding how these can be satisfied, as was the case for project 6; the stock control system, in the light of the client’s initial specification. For example, if a client wishes for a highly reliable system, then the design team may translate this into more specific requirements. As far as system functionality was concerned, only two projects used abstract structured representations (one JSD, and the other an in-house flow chart and screen contents document). Conceptual specification may involve nothing more than conversations and a few rough notes, or it may involve highly elaborate and detailed representations such as JSD process structure diagrams, mock-ups, lists of selected interactive methods and so on.

Where user requirements were concerned, informality was the general rule. However, one of the designers did try to adhere to strict usability principles. In some of the projects designers interviewed prospective users, or carried out informal task analyses by talking to and watching experts or potential users in the application domain, or by trying out tasks themselves to ‘get a feel for the job’ and come to a better understanding of what was required. The designer in project 7; the simulation training device "spent a lot of time with a subject matter expert" and found that he "had to get acquainted with the subject matter in order to do the design." In other cases designers had to work with more indirect information about users through descriptions in meetings or in requirements specification documents.

Such information about users, together with ideas from other products and advice from others involved in the design, would then form the basis for detailed design of UI functionality and operations. In the case of project 7; the simulation training device, which was to have non-computer expert users, it was decided to make it impossible for users to enter into a state from which they
would have difficulty escaping.

Conceptual specification appeared to be hampered in projects where the product was, in the designer's experience, novel. In project 3 it was difficult to imagine what the system might be like at all, especially since the application domain was unfamiliar to the designer. However the designer stated that he always aimed to make the system adhere to "a well formed metaphor" which would make it easier to learn.

This type of activity did not, in the projects studied, appear to involve any of the types of technique typically recommended in HCI DETs. For example no abstract dialogue specifications were generated, no user's conceptual models produced, no simulations of user processing or knowledge representations applied to system specifications. User-oriented design material came from managers, marketers, application domain experts, or in the best case direct from users themselves. None of this material seems to have been presented or analysed in a psychologically rigorous or structured manner. For example, the designer in project 4 (the educational graphics system) stated that users' experience of computers was important for example "it was relevant in deciding which keys caused what function", but the designer assumed that they would have time to learn the effects of the function keys, even if they were not ideally designed.

**Generation of a Working Prototype** seems to take up the most time and effort in development. All of the projects described prototyped the entire system and evaluated it, to determine the reliability and acceptability of the design. In seven of the projects it appeared that the design process centered on continual adaptation of running software, until it was deemed to meet the requirements specification. Projects tended to vary somewhat in terms of whether they adhered, more or less explicitly, to some principles such as sets of user-oriented principles (project 1), modular and modifiable code (project 2), block design (project 8), and so on.

Some of the products used early high level specifications of requirements or functionality which could be said to represent a top-down approach (for example JSD specifications were used in Project 6), however where such
specifications were used, they tended to be integrated with more informal and iterative generate and test regimes. The only project using structured, top-down design throughout, together with early use of specifications was the development of the simulation training device. This exception could be explained by the fact that the system had a highly constrained interface, with a small set of possible states.

What was clear for UI generation was that designers had many responsibilities and were always under pressure, so that some user-oriented measures tended to be pushed aside. For example the designer in project 5 stated that he "should have produced a tutorial guide [to the system] but never got round to it ... it was forever second down on the pile of things to get done". It may be that the breadth of responsibility designers have is the main reason for inadequacies in the usability of the interface.

Testing determines whether or not the UI satisfies requirements. Abstract evaluative descriptions of proposed interface characteristics were not generally used in the projects described. In particular, HCI DETs, as described in Chapter 2, were lacking altogether. In all cases any prototype testing was conducted or organised by software designers, possibly with the collaboration of a client themselves (i.e. it was not passed on to HCI specialists), directly on the prototype. It generally involved unstructured 'try it out' sessions, however in project 7 the requirements of the client were that user-evaluations be exhaustive and very structured, possibly because of safety requirements in its high risk industry. Some designers simply pretended to be naive users (the window manager designer from project 5 admitted to doing this).

User evaluation of a prototype appears to be more common as a source of information about users requirements than initial information gathering about users. Only one of the interviewed designers in the study carried out extensive user (the designer in project 3) tasks studies before starting to build a prototype. On the other hand 4 designers, including the one who also carried out extensive early task studies, (from projects 2, 3, 6 and 7) reported carrying out user evaluations, however informal. In the other cases prospective users were not involved in the design and evaluation of the system before it was finally implemented or marketed. This is not something which designers are happy
about. As the designer in project 7 put it "Whenever possible test things out... Always try to see the end-user because management maybe know too much and don't realise what the end-user knows or needs." All of the designers expressed similar opinions but it appears that not all of them had the time or support they needed to carry out appropriate evaluations. In other cases organisational factors may have been obstructive. For example, the designer in project 8 stated that he "had no contact with users because they wouldn't have bought the existing system". In other words the purchasers who spoke to the marketers in his company were not the users of the old release of the system and therefore could not provide direct feedback into the new design.

In the four projects where users were involved in prototype evaluation it was clear that many designers did spend considerable time and effort modifying the interface in response to any advice or ‘wouldn't it be nice if...’ suggestions they received. They also tried to find ways of 'crashing' the interface to make sure it was robust. However, evaluation of the developing UI was not rigorous, user performance goals and metrics were not used, and HCI specialists or specialists in psychological evaluation were apparently not consulted in the preparation of user evaluation regimes, if any took place.

**Finalisation** occurs when the benefits of improving the design *as a whole* are outweighed by the costs of putting off implementation or marketing. Designers interviewed were typically fairly critical of the final product, but justified shortcomings by explaining that they were unable, for various reasons, to improve it any further. This finding compares with the the findings of the features analysis where typical responses were that the system UI was satisfactory even though many constraints had been experienced. The garment design-aid designer was pressurised by the client into allowing it to be marketed long before that designer felt it was ready. He "saw the whole thing as a prototype [the marketing company] saw it as a product". This designer mentioned that there is sometimes a problem of modifications being made to a design after it has been handed over to a marketing client by a consultant software designer. The client may wish to make the design fit in with their product-range image, or make it possible to link up with other products. This may occasionally have a detrimental effect on the UI.
Typical development activities do not include abstract specification of a possible form for the UI. The kinds of activities described by interviewees tended to resemble those pinpointed in other design practice studies, being mainly devoted to developing and testing system functionality rather than the interface. Requirements for system functionality were ascertained during commitment to a design undertaking, and conceptual specification. Possible solutions were usually programmed in directly and informally tested, as described above. Consequently, the generation and evaluation of the UI appears to be an informal cycle of an iterative nature. However, in spite of designers’ good intentions, early involvement of users in the design process, as recommended by Gould and Lewis (1985) was a rare occurrence, and repeated user-evaluation during each iteration was only reported in a minority of projects.

The informality of various approaches to system development may encourage the trivialisation of the requirement for user-evaluation. None of the designers interviewed reported ever having used any HCI DET, some stated that they were unaware of the existence of such techniques. Notably, the finalisation of a system is frequently determined by external market pressures, rather than any design team satisfaction with the state of the UI. As long as the software does not crash and the system has the required functionality, system marketers may be happy to accept interfaces without user-evaluation.

5.3.4 Applied and Commercial Design Problems

The information volunteered by designers about problems they experienced in their projects was of great interest. It tended to suggest that failure in one area of a project tended to lead other problems in dependent areas. Table 5.1 lists problems which appear to be important in UI design, with the projects where they were most apparent. In the table they are listed in approximate order of first emergence. A brief illustration of each problem and its effects follows:

Poor Communication seemed to be the first common potential problem. In project 4, poor communication between the designer and the marketing client, meant that designer was unable to make use of the specifications they provided. In project 3 a lack of shared terminology between the designer and potential users made it difficult to communicate task characteristics and possible system functionality.
Designers' Unfamiliarity with the Task Domain increased the communication problems. The designer in project 2 reported a shortage of available information about users' tasks. In this project, the system application was a task which did not, as yet, exist so that its nature was difficult to determine. Also prospective users were not involved in the design and development process. The designer in project 3 found it hard to understand what the users' requirements for support were, due to lack of experience with pattern-design tasks, and had problems understanding pattern designers' explanations of their tasks. In project 7 a useful solution to this type of problem was found. The designer worked very closely with a subject matter expert who was able to provide valuable advice, and clear descriptions of the types of tasks the system would have to support.

Novelty of the Application was a considerable problem in projects 3 and 7. There were no existing comparable systems upon which to base ideas. The features analysis suggested that existing products were a significant information source for designers, without which it may be difficult to conceptualise the nature of the envisioned product and the pitfalls associated with it.

Uncertainty About Requirements for the design. In project 3, an explicit requirements specification was difficult to provide, so that the designer had to spend a long time finding out exactly what potential users wanted.

Exclusion of Users from the design process led to the designer in project 2 receiving completely misleading information about user-information requirements from a manager who was a system expert, unlike the end-users themselves. This resulted in an interface which was very unsatisfactory for the users who tried it out. Many modifications had to be made, which might have been avoided if users had been involved in the design from the early stages, as was recommended by Gould and Lewis (1985).

Complexity of the UI It seems that exclusion of users could lead to complex user interfaces. Projects 2 and 5 did not involve users before generating the prototype. In both cases significant user difficulties emerged as a result of the UI being too complicated; in the first case because the UI did not supply enough information to allow users to carry out their tasks (it assumed greater knowledge about the system than the users had) and in the second case the functionality was badly organised so
that mouse control and window manipulation was very difficult.

Expanding Task Outlines were common. Once users or clients saw a prototype in action, they often got ideas for extra, useful functions they would like, as well as the ones available. At least five of the projects involved in this study had to make significant changes in functionality throughout the design process. Designers from projects 3, 5, 6 and 8 mentioned that this was a problem.

Lack of HCI Guidelines and Standards meant that solution acceptability was left to the designers, or to clients, to evaluate. If they did not have experience with evaluation of usability, then this may not have been adequately considered. In project 5 and 8, design solutions were considered acceptable purely on the basis of their being 'bug' free and running quickly.

Familiar Solution Application was encouraged when there was pressure to complete a design quickly, rather than considering the specific nature of the particular project. In project 4 the designer was pressurised by a poor specification and lack of information about users into basing the design on past experience. In project 8 the designer had to base much of the system functionality on that of the existing old version, since users were not involved in the design process and their detailed requirements could not be determined.

There may also have been also a positive side to use of familiar solutions where good design ideas in comparable applications may propagate across the design community. This was indicated by evidence of a certain amount of "borrowing" going on. Designers in projects 1 and 5 were significantly influenced by existing systems, most of the others used experience from other products in some way. In project 3 the designer stated that he had some difficulty with the design because nobody involved had any experience of a comparable system.
Table 5.1.
Problems Experienced in Interface Design and Projects Where They Were More Apparent

<table>
<thead>
<tr>
<th>Numbered Design Projects</th>
<th>Problems in Approximate Order of Emergence in Design-Cycle</th>
<th>Design Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. <em>Display Editor</em></td>
<td>Poor Communication</td>
<td>2, 4, 8</td>
</tr>
<tr>
<td>2. <em>Network Management System</em></td>
<td>Designers' Unfamiliarity with Task Domain</td>
<td>2, 3, 7</td>
</tr>
<tr>
<td>3. <em>Garment-Pattern Design-Aid</em></td>
<td>Novelty of the Application</td>
<td>3, 7</td>
</tr>
<tr>
<td>5. <em>Window Manager</em></td>
<td>Exclusion of Users</td>
<td>2, 5, 8</td>
</tr>
<tr>
<td>6. <em>Office &amp; Stock Control System</em></td>
<td>Complexity of the UI</td>
<td>2, 5</td>
</tr>
<tr>
<td>7. <em>Simulation Training Device</em></td>
<td>Expanding Task Outline</td>
<td>1, 3, 8</td>
</tr>
<tr>
<td>8. <em>Distributed Building-Management System</em></td>
<td>Lack of HCI Guidelines &amp; Standards</td>
<td>5, 7, 8</td>
</tr>
<tr>
<td></td>
<td>Familiar Solution Application</td>
<td>4, 6, 8</td>
</tr>
<tr>
<td></td>
<td>Technological Limitations</td>
<td>1, 6</td>
</tr>
<tr>
<td></td>
<td>Written Software Limitations</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Over-Casual Evaluation</td>
<td>5, 8</td>
</tr>
<tr>
<td></td>
<td>Lack of Performance Metrics</td>
<td>2, 5, 8</td>
</tr>
<tr>
<td></td>
<td>Market Pressures</td>
<td>3, 5, 8</td>
</tr>
</tbody>
</table>
Technological Limitations were also a considerable problem for projects 1 and 6. The designers sought compromise solutions in both cases, since these limitations were unavoidable. In project 6, a major problem was caused by the fact that the editor on the PC could not be interrupted if a user wanted to move temporarily to another part of the system; if, for example, the user wanted access to information in the database during a phone-call, and that user was editing at the time, then they would not be able to do so. Compromise solutions would be to re-allocate tasks to different users, or to provide each user with two terminals.

Written Software Limitations could be frustrating later in the design, when seemingly trivial changes turned out to be impossible as a result of the way in which the software had been designed. If the task outline was still expanding later in the design then the system could not always meet it. In project 3 the pattern designers were excited by what the system could do and provided lots of ideas as to how it could be improved. However the designer was unable to implement these ideas. This designer actually stated that it would have been ideal to have been able to ‘go back and start all over again’. In project 2 the designer was lucky enough to be able to prepare for this eventuality and maintain a high degree of flexibility in the UI software which permitted him to make significant alterations to it late in the design cycle.

Over-Casual Evaluation was apparent in at least three cases. All of the projects, with the exception of project 7, carried out, at best, unstructured user-evaluations. At worst, designers pretended to be naive users themselves when testing the software in action. In project 5 a direct result of over-casual evaluation was an extremely overloaded window control icon which caused several different effects depending on the mouse click combination used to activate it. So a window could close unexpectedly, or disappear behind another one and so on.

Lack of Performance Metrics follows on from casual user evaluation. Evidence from Hammond et al (1983) suggests that users are not good at assessing the efficiency of their own performance. It seems probable that if users simply try out a system, they are unlikely to be aware of changes in their own performance, becoming engrossed in the novelty and powerful functionality of the ‘new toy’. Users’ initial opinions are not a satisfactory measure of performance.
Market Pressures were a notable excuse for weak UI design. They cannot be trivialised however as their impact on funding and time-scales is immense and often unavoidable. This encourages cheap solutions to problems or faster system response times, which may have a lot of selling power for a product. Projects 3 and 8 both suffered as a result of market pressures which in the first case forced an early end to design activities, and in the second restricted considerations to cost and system responses, rather than usability.

The problems described above appear to be highly dependent upon the nature of the design environment and activities. Pressure imposed by organisational, practical, and technological limitations and the exclusion of HCI oriented development activities in practice were all cited by designers as being responsible for the types of problems listed here. This kind of information was implied by the features analysis reported in Chapter 4.

Some problems may be avoidable, or at least reducible, if they are related to aspects of design which are largely in the designers’, or clients’ and developers control, particularly where there is commitment to proper user-oriented design practice. However other problems may be more unavoidable if they are outside the control of the designers or any other parties who have some influence over the design environment.

In effect design problems often represent the effects of constraints upon design activities. Sometimes they are a result of preexisting unavoidable conditions, or unforeseen conditions which emerge during the design process. Sometimes they arise as a result of avoidable errors of judgement, mistakes or, lack of awareness or consideration of certain issues. If unavoidable conditions exist in a commercial design project, it may be impossible to apply current HCI DETs. Although it may be possible for designers to adhere more closely to the kind of principles recommended by Gould and Lewis than they seem to currently, it may not be easy for them to resist external pressure which encourages them to do otherwise. If designers wish to resort to a prescribed HCI DET, they will need one that can be applied in conditions where the above problems exist. On the other hand, whilst remaining sensitive to unavoidable design constraints, HCI DETs must attempt to ensure that the avoidable ones are dealt with.
Finally, some reasons why HCI techniques are not applied were supplied by the designers themselves. Few of these designers were negative about the aims of HCI, the main problems were with the nature of the techniques and their presentation. These complaints should be taken as caveats for Human Factors specialists. They were as follows:

- No confidence in HCI as a discipline, and no perceived need for it.
- Lack of awareness or available information about appropriate techniques.
- HCI is seen to be too time consuming and expensive to be worthwhile.
- Techniques are often intimidating in their complexity.

These comments suggest that there is a certain amount of mistrust or misunderstanding of the discipline of HCI amongst some systems designers. Although such statements have been dismissed as unfounded, or as representing easily solved problems (e.g., by Gould & Lewis, 1985), the findings of this investigation suggest that there may also be practical reasons for not applying current HCI DETs.

5.4 Discussion of Interview Findings
With Respect to Questionnaire Findings

As stated earlier, the main goals of the study were as follows:
* To present a more integrated view of applied, commercial practice than was obtained by the features analysis of design projects. Design is viewed as a process where early decisions and activities influence later outcomes,
* To determine whether HCI DETs or principles are applied by non HCI specialists in commercial practice.
* To explain the design constraints and their effects which might prevent systems designers from applying more user-oriented design techniques.

In order to do this it has been necessary to study more closely the activities, main constraints and problems encountered by designers which might obstruct the application of good design principles and, more specifically, whether UI design reflects HCI DETs' design views.

5.4.1 Characterising Design Cycle Activities and Organisation

The interviews revealed additional information to that provided by the features analysis of design practice about the diversity of design environments. The
immense variability along many dimensions makes any detailed characterisation impossible, and it is likely that all such characterisations will either be idealisations or representative of only a small minority of design projects in general.

The interviews and the resulting examples of design scenarios provide a view of sequences of events and their organisation/relationships in typical design practice. Figure 5.1 outlines a tentative framework for the structure of typical design cycles, based upon the classes of design activity identified in the interviews. The five types of activity; commitment to a design undertaking, conceptual specification, generation of prototype, testing and finalisation are all theoretically optional, and can vary greatly in their formality and rigour. The only real fixed structure is that commitment, if it occurs, must come before generation which must precede finalisation.

In essence figure 5.1 is a minimal design schema which captures only those core aspects of design observed in this study which are relevant to the user (for example software issues such as defining abstract data types, core processes, and so on, as included in systems development methodologies such as JSD; Jackson, 1983, and SD; Yourdon & Constantine, 1978, are not included). For this reason it only represents the tip of the design iceberg, and many of the software implications of the activities it specifies are hidden. Even though it is intended only as a user-oriented design practice representation, this design schema still resembles Royce's waterfall model of design more closely than it does any of the design views presented by the HCI DETs described in Chapter 3. For example, although it bears surface similarities to the phased model of UI design described by Polson (1987), the detailed implications of each phase are completely different, these are elaborated by referring back to the design scenarios which represent a more typical set of design histories than proposed by Polson. It should be noted in particular that prototyping is more usually of the evolutionary type as proposed by Rosson et al (1987), than of the simulation type proposed by Kieras & Polson (1985).

However, there are still some important features, relevant to the applicability of HCI DETs, which are not included. Firstly, there are the more important design constraints identified in the features analysis and the main dimensions of variability. Added together with any common problems in the design environment these could represent obstacles to the application of HCI DETs. Secondly there are the information sources which designers typically exploit, and those which are typically
Figure 5.1
Framework for Common User-Oriented Design Cycle Activities
Based Upon Interviews with Designers

Commitment
Explicit agreement on general goals

Conceptual Specification
Defining detailed user-and functional requirements to satisfy general goals

Generation of Prototype
Generally Evolutionary
Building functionality to satisfy user requirements, verifying UI software

Testing
Validating functionality with respect to requirements

Finalisation
Maintenance and possible further evolution
inaccessible or inapplicable. We require a representation which can capture this variability.

HCI DETs must have realistic expectations of the information which designers should be able to exploit, and the user-oriented design activities they are likely to be able to undertake. If we are to produce a representative picture of design practice upon which to base such expectations, then common constraints, and information sources available must be acknowledged as they represent an important part of the design environment. They provide valuable information about whether design practice typically meets the application requirements of HCI DETs.

5.4.2 Problems as Manifestations of, or results of Design Constraints

Design problems are viewed here as frequently representing or resulting from constraints on design activity. The only possible distinction is that some designers may not acknowledge design problems as such, and prefer to think of them as constraints within which they have to work. The interviews tended to elicit many more admissions of errors of judgement, bad decision making and general difficulties than did the questionnaires. Permitting potentially avoidable constraints to remain, and perhaps stand as excuses for any failure in the design may be easier than attempting to solve them. This will be acceptable in design as long as clients, host organisations, designers and others involved all agree that certain problems are unavoidable.

The problems or constraints identified in this study can be compared with problems identified by respondents in the features analysis which were summarised in table 4.10 as of the following types:

- Technological problems
- Organisational
- Methodological
- UI complexity
- Poor design concept
- Pragmatic

Each of the problems identified in the interviews could be classed as an example of, or resulting from one of the above types of problem. In the features analysis it was clear that technological problems are the most prevalent, followed by organisational
and methodological, however the elaborations provided by the interviews give a clearer picture of the nature of such problems.

The problems identified here can be related to some of the types of constraints identified as important in the features analysis. The ten most important constraints from the features analysis are listed in table 5.2 (designers ranked these as the most important of 21 possible options). Some of the constraints were less common but ranked as more important where they did occur. This explains any discrepancies between the percentage incidence and the ranked importance of each constraint. Designers were invited to add and rank additional design constraints. However those that they added were mainly to do with the inadequacy of time and software and hardware resources. This suggests that inadequate resources may be a more important constraint than it appears to be in the ranking in table 5.2. In the following is a discussion of the way in which each of the top ten design constraints as identified in the features analysis can be seen to have emerged in the projects of the interviewees, but more interestingly the interviews provided better explanations of their roots, and details as to their effects.

**Complicated application/product sophistication** placed greater demands on the designer. Designers seemed to find novel applications very complex, with uncertainty about how to approach the design in general and a poor picture of possible solutions. Projects 3 and 7 experienced problems with the complexities of the application domains with which they were unfamiliar, and with the applications themselves which were novel and quite sophisticated. This constraint led to uncertainty about requirements, in project 3 expanding task outlines resulted. In project 8, involving a relatively sophisticated product the designer had to rely on familiar solution application. For many of the designers technological and written software limitations (e.g. projects 1, 3, 4, and 6), and market pressures (especially 3 and 6) seemed to limit what was feasible and added to the complexity of solving design problems.

**Lack of information about users** was caused by poor communication and cooperation with clients and users or exclusion of users (e.g. projects 2, 3, 4, 5 and 8). A general lack of HCI guidelines and standard usage may have made this problem worse. It led to a number of problems, including uncertainty about requirements, complexity of the UI, familiar solution application, over-casual evaluation, and lack
Table 5.2
The Ten Most Important Design Constraints
Identified in the Features Analysis of Design Practice

<table>
<thead>
<tr>
<th>Rank</th>
<th>Design Constraint</th>
<th>Weighted Importance</th>
<th>Percentage Incidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Complicated Application/</td>
<td>149</td>
<td>79.2</td>
</tr>
<tr>
<td></td>
<td>Product Sophistication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td>Lack of Information about Users</td>
<td>74</td>
<td>41.7</td>
</tr>
<tr>
<td>2.5</td>
<td>Over-casual Approach to Evaluation</td>
<td>74</td>
<td>41.7</td>
</tr>
<tr>
<td>4</td>
<td>Inadequate Resources</td>
<td>71</td>
<td>45.8</td>
</tr>
<tr>
<td>5.5</td>
<td>Lack of Assistance/</td>
<td>60</td>
<td>33.3</td>
</tr>
<tr>
<td></td>
<td>Collaboration from Client</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.5</td>
<td>Lack of Experience with Interface Design</td>
<td>60</td>
<td>33.3</td>
</tr>
<tr>
<td>7</td>
<td>Undersized Team</td>
<td>48</td>
<td>37.5</td>
</tr>
<tr>
<td>8</td>
<td>Lack of Guidance From Parties Outside of Team</td>
<td>46</td>
<td>29.2</td>
</tr>
<tr>
<td>9</td>
<td>Lack of Experience With HCI</td>
<td>40</td>
<td>29.2</td>
</tr>
<tr>
<td>10</td>
<td>Over-casual Approach to Design</td>
<td>38</td>
<td>25.0</td>
</tr>
</tbody>
</table>

of performance metrics.

Over-casual Approach to Evaluation seemed to be a result of the designers’ lack of familiarity with HCI techniques, many of them knew little or nothing about
user-oriented evaluative methods such as those recommended by Jorgensen (1984). In project 5 this led directly to a complicated UI. Most projects did not have any clear performance metrics in mind when evaluating their systems. Only project 7 used a rigorous and structured user evaluation regime, where users had to be able to reach a criterion level of performance in terms of UI errors, and their subjective assessments were directly probed by a post evaluation questionnaire. For those projects where evaluations were not rigorous it seems probable that many difficulties were never identified by the designers.

**Inadequate Resources** were a frequent complaint in both the features and the interview studies (e.g. projects 3, 4, and 6). In particular lack of time seemed to be a major grievance with designers in the features analysis spontaneously listing it as a general constraint in its own right. With better resources designers felt that they could have got better hardware and software. They would have found ways of avoiding problems, such as lack of information about users and tasks (e.g. project 5), and familiar solution application (e.g. project 8).

**Lack of Assistance/Collaboration from Client** was cited as a major constraint in 3 out of the 4 projects where a client organisation was involved. In particular in project 2 the client did not allow the designer to work with potential users to get a good picture of their requirements in the early stages of the project this led to exclusion of users from the design process, uncertainty about requirements, and complexity of the UI. In project 4 the designer's client provided a very poor specification and was not cooperative with attempts the designer made to make it more workable, the designer was left guessing somewhat about user and client requirements and had to rely on familiar types of solution which he was confident would work.

**Lack of Experience with Interface Design** was admitted to by the designers in projects 5, 6, 7 and 8. This is not to say that they were not experienced software writers. However they felt unsure about the kinds of requirements users might have because they did not have enough experience with user feedback from similar solutions they might have used in the past (and as the features study suggested, designers tend to rely heavily on past experience as an information source for UI design).

**Undersized Teams** were the rule, rather than the exception in this study. The
majority of the designers worked largely alone with little support. In project 7 the designer was lucky enough to be able to work with a subject matter expert to guide design decisions, the features analysis suggested that working with application domain experts and consultants with various areas of expertise is a valuable aid to designers. However in this interview study most of the designers had to manage alone. In project 3 the designer was heavily pressurised and unable to satisfy many of the targets he had hoped to. He also had to educate himself in the application domain with which he was unfamiliar. In project 4 the designer felt isolated from any assistance, in project 6 the designer found great difficulty in solving technical problems caused by inadequate software and hardware. Essentially any problem which emerges in a smaller team is likely to be harder to solve. However the designer in project 1 was pleased to work alone because he had experience of many problems which can emerge with larger teams (such as lack of shared goals, inconsistencies in the system, and so on).

**Lack of Guidance From Parties Outside of Team** seemed to be linked to lack of assistance and collaboration in project 2 where the designer found that the specification was no use in directing the design because it was unworkable. On the other hand, in project 3 the client manager was willing to help but lacked the necessary knowledge. He knew little about the application domain, and even less about what functionality to suggest a CAD system might provide, and the kind of tasks it might support, communication problems, uncertainty about requirements and an expanding task outline resulted. The main problem relating to this constraint was uncertainty about requirements of the client.

**Lack of Experience With HCI** was a problem for several of the designers, mainly those in commercial organisations. One was very familiar with HCI, another who was quite familiar with it stuck to a number of user-oriented design principles, two worked very closely with users and carried out repeated iterations, and another worked closely with a subject matter expert and carried out rigorous evaluations. However projects 4, 5, and 8 where the designers were not experienced with HCI and did not work closely with users, many problems such as uncertainty about requirements, exclusion of users, complexity of the UI, over-casual application, and lack of performance metrics resulted.

**Over-casual Approach to Design;** the least common design constraint in the
features analysis, was rarely admitted to in the interviews, but was indicated in particular by the designer in project 5. After the initial involvement of a UI design consultant, the system was informally and iteratively generated and tested, but no user evaluation was carried out, and there did not appear to be very good reasons for this. The design team never looked at user requirements beyond what they considered would be potentially useful functionality and what were available ways of providing it. Prospective users were excluded throughout the design, and the designer informally tested the UI by pretending to be a naive user. The resulting UI was not satisfactory in the designer’s view. A possible reason for not including prospective users in the design process was that they were a diffuse group in the general public, this effect seemed to occur for project 4 also. In all the other projects, prospective users were easily identifiable, and accessible, employees of various design host or client organisations. In all the other projects prospective users were at least involved in prototype evaluations.

The above illustrations suggest that design constraints tend to act as problems or to lead to problems for the design team, for example poor communication and exclusion of users lead to uncertainty about requirements. This information was not made clear by the features analysis, based on questionnaires. Some of the problems or design constraints described by interviewees may be avoidable or at least reducible in some circumstances if they are related to design aspects which are largely in the designers’, clients’ or design tool developers’ control, for example exclusion of users who may not be an easily accessible target group, should not occur if they are likely to be easily contacted members of the general public. For projects 4 and 5 it seems hard to believe that representatives of user groups; children and computer workstation users respectively, could not have been quite easily and cheaply accessed, at least for product evaluation (they could even have been friends of the designers, or their children).

If HCI techniques and user-oriented design constraints emphasise the causes and effects of these problems and provide methods for avoiding them, it is likely that design practice will benefit, particularly where there is commitment to proper user-oriented design practice. Avoiding design problems or constraints, particularly those which emerge early is likely to reduce the number of problems emerging later on, simply because so many of these problems are causally interrelated.
Table 5.3
Design Constraints and Problems Combined
from the Features Analysis and the Designer Interviews

<table>
<thead>
<tr>
<th>From the Questionnaire-Based Features Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complicated Application/Product Sophistication</td>
</tr>
<tr>
<td>Lack of Information about Users</td>
</tr>
<tr>
<td>Over-casual Approach to Evaluation</td>
</tr>
<tr>
<td>Inadequate Resources</td>
</tr>
<tr>
<td>Lack of Assistance/Collaboration from Client</td>
</tr>
<tr>
<td>Lack of Experience with UI Design</td>
</tr>
<tr>
<td>Undersized Team</td>
</tr>
<tr>
<td>Lack of Guidance From Parties Outside of Team</td>
</tr>
<tr>
<td>Lack of Experience With HCI</td>
</tr>
<tr>
<td>Over-casual Approach to Design</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>From the Systems Designers’ Interviews</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poor Communication</td>
</tr>
<tr>
<td>Designers’ Unfamiliarity with Task Domain</td>
</tr>
<tr>
<td>Novelty of the Application</td>
</tr>
<tr>
<td>Uncertainty About Requirements</td>
</tr>
<tr>
<td>Exclusion of Users</td>
</tr>
<tr>
<td>Complexity of the UI</td>
</tr>
<tr>
<td>Expanding Task Outline</td>
</tr>
<tr>
<td>Lack of HCI Guidelines &amp; Standards</td>
</tr>
<tr>
<td>Familiar Solution Application</td>
</tr>
<tr>
<td>Technological Limitations</td>
</tr>
<tr>
<td>Written Software Limitations</td>
</tr>
<tr>
<td>Over-Casual Evaluation</td>
</tr>
<tr>
<td>Lack of Performance Metrics</td>
</tr>
<tr>
<td>Market Pressures</td>
</tr>
</tbody>
</table>
Some problems are often related to design constraints which are outside the control of the designer, such as time, resources, lack of assistance from the client, for example technological limitations caused by inadequate resources for the project may simply have to be endured, no matter how many problems result. Future design techniques must be sensitive to the extra demands and the limitations placed by these problems upon all concerned in the design process.

The findings relating to design constraints and problems from the features analysis and the interviews are combined in table 5.4. Some of these overlap, as one would expect, but others are revealed as a result of the differences between the two studies. They show some of the problems, avoidable and otherwise which designers have to contend with in applied practice as opposed to artificial test beds for psychological and HCI techniques and tools. They are the kinds of problems which HCI DETs must address if they are to be more widely applicable.

Table 5.3 lists design-organisational and technological limitations, designers' lack of experience with UI design and with the system's task domain plus many other factors which may represent real problems for interface designers. Even if a designer is highly supportive of HCI, these factors may make it impossible to apply DETs successfully to commercial design in general. Hammond et al (1983) suggest that, rather than supplying detailed guidelines, Human Factors specialists should perhaps concentrate on more appropriate 'mini-theories' of the user and user-performance for designers to use. What appears to be certain is that the design and development process is unpredictable, with a large number of possible constraints which may emerge. Designers need a wide variety of flexible DETs which can satisfy constraints of the kind identified here. Observations from design studies indicate that HCI DETs must produce useful, comprehensible information and be easy for the non-HCI specialist to apply, since HCI specialists may be a rarity in commercial design projects. In Chapter 7 the implications of these constraints for the Application of HCI DETs in design will be discussed.

5.4.3 Exploitation of User-Oriented Information Sources

The features analysis revealed that the number and type of user-oriented information sources exploited by systems design projects varies widely. However some sources are markedly more popular than others. Table 5.4 shows the ten most
important user-oriented information sources as ranked by designers out of 18 possible choices in the questionnaires in the features analysis. In addition to these designers added personal experience, specialist expertise and existing comparable systems, which they ranked as being of great importance. In the following, these findings are compared with those from the interviews and design scenarios they generated. The interviews provided more qualitative and detailed information about the way in which user-oriented information sources are typically exploited.

Specifications of To-Be-Supported Activity proved to be crucial in the projects studied. They were supplied to designers by clients in projects 2, 4 and 7. In projects 3, 5, 6 and 8 the designers or design team collaborated with clients or with specialists to produce specifications. In project 1, which was the only non commercial project, the designer developed his own specifications. In all eight projects the specification served as a focus for design. In project 4, where the specification was ill conceived, the designer changed it himself, and this case illustrates the importance of the specification and some of the problems of inadequate specification. The designer had considerable difficulty ascertaining what the client’s and users' requirements were and had to enter into conflict with the client in order to make the necessary changes.

Observation of Prospective User Activity was very valuable to designers wherever it was possible. In projects 4, 5 and 8 designers recognised that the lack of this information source represented a major weakness in their design, however it does not seem to have occurred to them that non-design team members would have been a possible second best; nobody mentioned settling for evaluation with individuals who might represent naive users. The other projects used prospective users to evaluate their prototype in iterative cycles. Most evaluation sessions were unstructured and informal. Only projects 3 and 6 involved observation of users’ activities before beginning prototyping.

Observation of Activity Using Prototype was crucial as an evaluative aid. All of the projects (1, 2, 3, 6 and 7) which observed use of their prototype did so with the involvement of prospective users. It is probably quite rare for designers to ask non-prospective users to evaluate the UI when no prospective users are available. Where it was impossible to obtain prospective users, designers tended to test the
### Table 5.4
The Ten Most Important Information Sources Identified in the Features Analysis of Design Practice

<table>
<thead>
<tr>
<th>Rank</th>
<th>Information Source</th>
<th>Weighted Importance</th>
<th>Percentage Incidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Specifications of To-Be-Supported Activity</td>
<td>113</td>
<td>52%</td>
</tr>
<tr>
<td>2</td>
<td>Observation of Prospective User Activity</td>
<td>108</td>
<td>56%</td>
</tr>
<tr>
<td>3</td>
<td>Observation of Activity Using Prototype</td>
<td>107</td>
<td>64%</td>
</tr>
<tr>
<td>4</td>
<td>Documentation on Related Activities</td>
<td>86</td>
<td>44%</td>
</tr>
<tr>
<td>5</td>
<td>Verbal Task Descriptions from Current Activity Performers</td>
<td>84</td>
<td>40%</td>
</tr>
<tr>
<td>6</td>
<td>Interviews with Prospective Users</td>
<td>76</td>
<td>40%</td>
</tr>
<tr>
<td>7</td>
<td>Experimentation with/Testing of Prospective Users</td>
<td>74</td>
<td>52%</td>
</tr>
<tr>
<td>8.5</td>
<td>Scientific/Psychological Ref's on Human Behaviour etc</td>
<td>56</td>
<td>36%</td>
</tr>
<tr>
<td>8.5</td>
<td>Verbal Task Descriptions from Other Persons</td>
<td>56</td>
<td>28%</td>
</tr>
<tr>
<td>10</td>
<td>Observation of Activity Not Using Prototype</td>
<td>38</td>
<td>24%</td>
</tr>
</tbody>
</table>
system themselves, attempting to crash the software, or reveal inputs which the system could not deal sensibly with.

**Documentation on Related Activities** was used by projects 1, 3, 5 and 8. The use of such material helped to provide examples of existing methods of providing the required functionality, and sometimes more information about the application domain tasks.

**Verbal Task Descriptions from Current Activity Performers** were supplied by prospective users who were already performing tasks which the system was to support in projects 3 and 6. The designers in these projects spent considerable time with the prospective users and relied heavily on them for both task information which directed system functionality and the nature of operations users would perform.

**Interviews with Prospective-Users** were not observed as being used by the designers in this study. The general rule appeared to be informal discussions or meetings with prospective users. This could also have been true for the respondents in the features analysis who cited this as an information source, unfortunately the questionnaires did not reveal if this was the case.

**Experimentation with/Testing of Prospective-Users** was only used in project 7, where a rigorous evaluation regime was devised by the design team’s client. The high number of features analysis responses to this information source could have been due to respondents informal definition of what constitutes experimentation and testing, which, if the interviews here are representative, may have been nothing more structured than observations of, and discussions with, users who tried out the prototype.

**Scientific/Psychological References on Human Behaviour etc** were not extensively used by any the designers interviewed in this study, although some had experience of such literature. They tended to give reasons such as those described earlier relating to the inapplicability or complexity of the material available.

**Verbal Task Descriptions from Other Persons** were used in project 7 where the designer worked closely with a subject matter expert who had already learnt the
skills which the prospective users would be trained in by use of the system. In projects 2 and 3 the designers were given task descriptions by managers in their client companies. However these proved to be inaccurate and either vague or misleading.

Observation of Activity Not Using Prototype Only projects 3 and 6 involved observation of prospective users tasks without system support. Project 2 involved use of a mock-up but those involved were not prospective users.

For the three information sources added by respondents in the features analysis, the following describes the kind of role they played in UI design in some of the projects studied here.

Personal Experience of previous design projects and use of comparable systems played a major role for all of the designers interviewed. The designer in project 7 stated that his lack of experience meant that deadlines were loosened, and extra supervision and help given, in recognition of the difficulties his inexperience might give rise to.

Specialist Expertise was contributed in projects 2 (from management), 3 (from industrial experts), 5 (from an HCI/UI consultant), 7 (from a subject matter expert) and 8 (from specialists in different fields within the company). Their input was very varied depending upon their area of expertise, generally influential, and in all except project 2, appreciated by the designers.

Existing Comparable Systems were studied by the designers in projects 1, 4, and 5. In project 8 the existing system, which was to be updated was used as a basis for the new version. In projects 3 and 7 the designers stated that the absence of existing comparable systems made design more difficult because it was hard to envisage the nature of the planned system and its UI.

The findings here tend to confirm those of the features analysis in that an informal approach to user-oriented design is typically favoured by designers. Informal prototyping with prospective users seems to be the most popular route to usable systems in current design practice. Users are less likely to be involved in the early stages of design than later on, and their input seems to consist of unstructured try-it-out sessions rather than rigorous experimental studies. However, it may be that the very
nature of casual use of a prototype may be informative in other ways. In project 3
the designer was surprised to see a user begin using the system for tasks he had not
built it to support. This revelation would probably not have emerged with struc-
tured experimental evaluation but it had a major influence on later design.

5.4.4 Characteristics of Commercial Interface Design Practice and the
Non-Use of HCI DETs

Although considerable time and effort were frequently directed towards planning
and evaluating UI designs, this effort seemed to be applied in very haphazard, infor-
mal ways. There is some agreement with the findings of the design study by Ham-
mond et al (1983) in that, although the designers were not ‘computer-centric’
(meaning designers did not only consider the system and ignore its functions, users
and environment), their approach often appeared to exclude important user require-
ments. Two projects were observed to have excluded users from the design process,
possibly because they were not a clearly defined easily accessed group in one of the
organisations concerned with the design. If designers were more aware of the
benefits of user involvement, they could be in a better position to convince other
interested parties that user involvement, even though it might take some extra time,
is important.

The interviews indicated that the three principles espoused by Gould and his col-
leagues were not applied, despite positive attitudes held by designers. Some
designers spent a great deal of time engaged in user- and task-oriented investigation
and development. This kind of activity was only possible when users were accessi-
ble. Since users were sometimes hard to approach, it is not surprising that early
focus on users was sometimes difficult. In some cases users were excluded alto-
gether, or included only at the latest stages in development. Designers' empirical
measurement was particularly weak; proper performance metrics and goals were not
used. It is probable that much better evaluation would be possible if designers were
able to characterise interfaces in such a way that testable performance goals could
be generated.

Finally iterative prototyping was most commonly incremental development (as
described in the design study of Rosson et al, 1987) where the prototype is the
developing implementation itself. However this prototyping often did not include
user-testing in its cycle. This meant that as far as usability was concerned, the value of iteration was often lost. This finding is very much in line with that of Rosson et al (1987) who found that projects taking iterative design approaches were no more likely to carry out early user evaluations than projects using phased development where the system is more fully specified before implementation takes place. Most notable was the fact that none of the designers interviewed used HCI DETs (they had either not heard of them, or they gave specific reasons for not using them). It therefore appears that this investigation has fulfilled its second aim in determining whether or not HCI DETs are used.

A view of design as practised is represented by the framework in figure 5.2, which is based upon the findings of the features analysis reported in Chapter 4 and those from the interviews with the eight systems designers. The ten most important user-oriented information sources and design constraints are represented in the figure, in approximate order in relation to the design activities. It should be noted that this figure, like figure 5.1, relates only to user-oriented aspects of design, and the software and hardware oriented, or more computer-centric aspects of design are only implied by some of the constraints, namely: Lack of Experience with Interface Design (possibly implying difficulties with implementation of the interface software), Complicated Application/Product Sophistication (implying problems with the application software and general complexity of the system as a whole, which was generally the most important constraint identified in the features analysis), Inadequate Resources & Undersized Team (implying too little in the way of time, money, advanced technology, and human-power which must all be distributed over the entire product). These possibly computer based constraints may represent the major drain away from user-orientedness in the focus of the design. Smith and Mosier (1984), on the basis of a survey, estimated that 30 to 35 percent of the software in an application is designed to support UI functions. This leaves 65 to 70 percent of the software to be designed has little impact upon user oriented aspects of the system, other than on its scope (i.e. the functionality it supports). It may be easy for systems designers to forget that some of their software, if implemented in a certain manner, has the potential to compensate system users, when much of it does not.

It should perhaps be emphasised that HCI probably has an important role to play right from the very earliest moments (Gould & Lewis 1985) through to the end; for
example a system may be supplied with performance predictions based upon the final version (Card et al 1983), or it could be supplied with an HCI based explanation of all the design decisions contributing to its final form as an aid for maintainers and users of the system (Maclean et al 1989).

If they were developed in such a way that systems designers found them easy to use HCI techniques of the future might provide methods for improving the rigour of requirements specifications, user conceptual and task oriented models of the system, and evaluative methods. At present it seems that some designers are happy to implement first and think afterwards and the features analysis suggests that there are occasions where designers are not clear as to what the aims of their evaluations really are. They need techniques which can focus their attention upon users needs early on, and which encourage them to use these needs as guidance throughout design and in the evaluation to ensure that they are really satisfied by the UI.

5.5 Conclusions

The main goals of the study were as follows:
* To present a more integrated view of applied, commercial practice than obtained by the features analysis of design projects.
* To determine whether HCI DETs or principles are applied by non HCI specialists in commercial practice.
* To explain the design constraints and their effects which might prevent systems designers from applying more user-oriented design techniques.

As a supplementary study this interview analysis has confirmed and enhanced the findings of the previous questionnaire based features analysis. Most importantly the lack of use of HCI DETs, as described in Chapter 2, is again highlighted as is the preference for in-house or unstructured iterative development approaches, rather than SADMs such as JSD (Jackson, 1983) and SD (Yourdon & Constantine, 1978).

A general schema for design practice has been presented, describing UI design as a series of goal-oriented activities which relate to a number of information sources and are subject to a number of design constraints. The designers in the study tended on the whole to recognise the importance of usability. One designer did apply some
Figure 5.2
Schema for User-Oriented Design Practice: Organisation, Constraints and Information Sources Based Upon Findings of Two Empirical Studies

<table>
<thead>
<tr>
<th>INFORMATION SOURCES OBSERVED</th>
<th>DESIGN ACTIVITIES OBSERVED</th>
<th>DESIGN CONSTRAINTS OBSERVED</th>
</tr>
</thead>
</table>
| Specifications of to-be-supported activity | Commitment
Explicit agreement on general goals | Lack of experience with interface design |
| References and documentation on related activities | Conceptual Specification
Defining detailed user- and functional requirements to satisfy general goals | Lack of experience with HCI |
| Observation of task activity | Generation of Prototype
Generally Evolutionary
Building functionality to satisfy user requirements, verifying UI software | Complicated application or product sophistication |
| Verbal task descriptions and interviews with users experts and others | Testing
Validating functionality with respect to requirements | Lack of information about users |
| (Possibly Parallel) | | Inadequate resources and undersized team |
| Observation of use of prototype | Finalisation
Maintenance and possible further evolution | Over-casual approach to design |
| | | Lack of assistance or collaboration from client |
| | | Lack of guidance from parties outside of team |
| | | Over-casual approach to evaluation |
user-oriented design principles, others tried to involve prospective users in the actual design process, or carried out empirical user evaluations.

A minority seemed to evade addressing usability seriously but acknowledged that this was not ideal. In spite of a generally positive attitude towards HCI, some negative statements were made about the usefulness of HCI techniques to design practice. These comments reflect the fact that not one of the designers interviewed in this study, or responding to the questionnaire in the previous study reported using one of the influential (within the discipline) HCI DETs described in Chapter 2. Reasons for not doing so appeared to be largely to do with the nature of constraints under which designers seem to operate. Lack of adequate resources, design difficulty, lack of information about users and lack of experience in psychology may be some of the most serious obstacles to application of HCI DETs in applied and commercial design practice.

It is worth emphasising the fact that designers do not make a clear separation between the UI and other factors such as the application and the target tasks when considering its usability. The functionality of the system, the modifiability of the software, unforeseen tasks and so on were considered as important to usability by certain designers. Their breadth of view is something of a vindication of the scoping matrix used in Chapter 2 as a useful way of viewing HCI techniques with respect to their applicability. The scoping matrix emphasizes consideration of a number of factors which impact upon usability, on the assumption that design or evaluation which restricts itself to consideration of only one factor (such as the user) and ignores others (such as the target tasks) is likely to prove unsatisfactory. This suspicion has begun to be raised by other researchers in HCI (e.g. Grudin 1989).

The main contribution this study makes to the overall process of user-oriented design practice is that it enriches our view of design practice as a series of causally related activities and presents explanations for design constraints and their problematic effects, and further descriptions of how designers typically exploit user-oriented information sources in applied and especially commercial systems design practice.

Unfortunately, the absence of any attempted application of any HCI DET still leaves a rather large gap in our understanding of whether or not HCI DETs are
applicable to commercial UI design practice. It is possible that designers' natural suspicions, as evinced by their complaints about HCI, really only related to the necessary investment in time. If only HCI specialists in well resourced organisations are using HCI DETs as a matter of course, then the explanation for under-use of these techniques may well lie with their requirement for time. If they find them difficult to interpret and rely on their training in psychology and HCI, then there may be a requirement for specialist expertise which systems designers do not generally have. However, if even HCI specialists do not use these techniques, then other explanations may be required for their lack of use. Such issues are the focus of the study reported in the following chapter.
6.1 Introduction and Rationale

This chapter describes a study into the practice and techniques applied by HCI specialists in commercial organisations. In the first part of the chapter the organisations employing the specialists, the working regimes, roles and activities of the HCI practitioners are described. Then the techniques which they applied within these organisations are described in some detail together with their advantages and their problems.

In Chapter 3 it was noted that there appears to be strikingly little in the way of literature or reports on the use of HCI DETs in design practice, particularly in the hands of other analysts than the developers of the techniques themselves. The study reported here represents an attempt to present some information relating to the use of HCI and user-oriented techniques in commercial design practice.

In the preceding two studies interface design practice in applied, and especially commercial, projects was observed and analysed in terms of its features and general organisation. However, precise techniques applied to assist in UI design practice were not identified. The main reasons for this were that in the features analysis the data collection method (i.e. questionnaire) was not well suited to detailed discussion, by respondents, of particular methods they used, and in the supplementary, interview-based study it appeared that little use was made of structured methods and there were no examples of explicitly user-oriented, structured methodologies being applied.

The features analysis and interview study of UI design practice indicated that, in general, systems designers do not tend to use the kinds of technique for design and
evaluation described in Chapter 2 (HCI DETs). On the other hand the features analysis suggested that large projects with HCI specialists were much more likely to exploit user-oriented information sources, such as experiments or informal evaluations of prototypes with users, task specifications (or to-be-supported activity specifications) and interviews with prospective users.

A further study was undertaken in order to look at explicitly user-oriented UI design approaches, in order to see what characterises an applicable HCI design or evaluative technique might possess. In order to obtain this information HCI specialists were interviewed as it seemed highly likely that such individuals would be more likely to apply HCI DETs of the kind discussed in Chapter 2.

This detailed interview study aimed to provide more precise information about the nature and scope of user-oriented techniques in commercial practice than was captured by the previous studies. The previous studies served to provide a view of various features of UI design and a picture of the design as a process of activities and repercussions. However, they did not provide sufficient detail relating to the use of user-oriented techniques in particular. It was hoped that interviews with HCI practitioners would be more likely to reveal this kind of information, as their roles in design should be much more likely to foster use of such techniques.

The rationale supplied a number of clear objectives which the interviews were intended to achieve. These aims are summarised by the following questions which were to be addressed:

* What precisely are the HCI specialist’s roles in the commercial design process.

* What design and evaluative techniques do they use, what is the scope of each of these techniques, and how are they used.

* How do HCI specialists using user-oriented techniques avoid or cope with user-oriented design constraints and problems.

* How do user-oriented techniques exploit information sources.

* How do the techniques support the activities of the HCI specialist, and in what
ways do they fail to do so.

The targets of the study were individuals with education and experience in psychology, ergonomics and/or HCI employed by medium to large, well resourced organisations. Even more than with the systems designers in the previous study, candidate interviewees employed specifically for commercial projects, as opposed to research, were hard to find. Six HCI specialists from four different organisations agreed to take part. Two interviewees from one organisation were unable to provide more time than an hour each, however the other four were able to provide two or more hours and a great deal of detailed information.

It might be possible that the main reason for the under use of HCI DETs reported in the studies in chapters 4 and 5 is purely to do with the heavy requirement they have for psychological and HCI expertise on the part of the analyst and also with the investment of time and effort needed to apply them. If this were true we might expect to see HCI specialists in well resourced organisations making much greater use of such techniques than do systems designers in general.

HCI specialists were also considered to represent a valuable information source in terms of their alternative roles in the design process. Unlike the interviewees in the previous study who generally had to address the entire system design (including the application end of the software and other interests such as system response time, security of software, memory requirements and so on) the HCI specialist concentrates on a subset of design issues. In order to carry out his or her tasks, the HCI practitioner will have to gather a great deal of information from and communicate with prospective users and other specialists or generalists who are unlikely to share the same background and skills. In other words there is not only a question of techniques used, and their applicability from the HCI specialist's point of view. There is also the issue of the HCI specialist's activities within system design as, for example, an analyst a collector and a communicator of specialist information. Of interest here is whether successful UI design and evaluative techniques used by HCI specialists support one or more of these activities.

From Chapters 2 and 3 it should be clear that HCI DETs are typically incomplete in terms of the extent to which they specify their own application and interpretation. This feature is, in effect, responsible for their requirement for psychological exper-
tise which enables the analyst to judge, for example, the appropriate grain of analysis at which to describe actions which fulfill goals, subgoals and even sub-subgoals. The same features which lead to uncertainty, in terms of the correctness with which the technique is being applied (for example, with TAG, an analyst might be unsure as to what constitutes a simple task) also allow a degree of flexibility which permits the analyst to use the approach in a way which might have been unforeseen or unspecified by its authors.

In addition to this it is possible that HCI DETs could, in many ways, be extended to cope with new applications. Moran (1981), for example, suggests possible extensions to the CLG including cognitive architectures supplying relevant human processing constraints which could permit performance metrics to be derived from a CLG specification. Knowles (1989) has investigated qualitative extensions to CCT in terms of the knowledge represented by the goal hierarchies in the user's how to do it knowledge. It seems likely that HCI specialists may adapt techniques to suit their requirements in various projects with different demands.

There is also the possibility that very different techniques are exploited by HCI specialists practising in commercial environments to provide the same kind of information supplied by HCI DETs developed in research environments. These might well be idiosyncratic, but the important fact is that the motivation driving their application, in addition to capturing and analysing relevant aspects of users, interfaces or interactions, is likely to be that the technique has practical value for the analyst in commercial design, rather than value in terms of its furthering understanding of theoretical issues. In other words such techniques have to be justified in terms of the value they add to design over the effort and cost they incur for the analysts who work with them.

6.3 Interview Structure and Methodology

An important difference between the stance adopted in the HCI specialist study and the two previous studies was that the specialist interviews were not restricted to a single project. The interviewees were encouraged to focus on particular methods rather than particular projects, and, time allowing, they were also asked to discuss a number of more philosophical topics relating to their experience of design in general. The reason for this difference was the new focus which concentrated upon de-
tails of techniques rather than on the whole design process. It was considered important to discuss as many as possible of the HCI specialists’ experiences with each particular technique covered. The focus of the interviews also encouraged the HCI specialists to describe any relevant experiences they had with previous employers. As it turned out, none of the interviewees described using more than two well structured, theoretically based techniques, although most of them exploited a number of empirical, informal or heuristic methods, including user observation, experimental user trials and interviews.

The interviews were structured around the following stages:

Stage 1.

1. Interviewee describes own background with respect to HCI and computer system design experience

2. Interviewee describes organisation(s) within which s/he practices/d

3. Interviewee describes Team structure (plus roles) within which s/he works/d

Stage 2.

1. Interviewee lists as many as possible HCI or SAD Techniques applied, successfully or otherwise

2. If possible interviewee describes the relative usefulness of each technique.

3. Starting with the most widely used, the interviewee discusses each technique in turn. For each technique interviewee describes situations within which it was applied in terms of a number of aspects (see Appendix 4)

4. Interviewee describes any constraints on design activities experienced.
5. Interviewee describes information sources used during the various design processes and states how these were exploited.
Stage 3.

Interviewee discusses design philosophy and issues more generally.

The structure of the interview was sensitive to the descriptions of the interviewees, such that they were not asked to answer questions on issues which they had already covered earlier. All of the interviews were taped for later analysis. The amount of detail required regarding the precise nature of each of the techniques the specialists used made it appropriate to prepare detailed word-for-word transcriptions of each interview (these interviews had to provide as much detail on HCI practitioners as the two previous studies combined had done on systems designers). The transcriptions were then analysed to determine various features of, and similarities and differences in, the working practices of the various HCI practitioners.

6.4 Findings

The findings are structured, together with a certain amount of discussion, around the issues addressed by the interviews and the objectives and targets of the study. The first aim of the study was to determine what precisely are the HCI specialist’s roles in the design process. This aim was supported by the focus of the interview in Stage 1 which covered the experience of the specialist, the nature of host organisations employing that individual, and the team structure and the roles within which he worked (all of the interviewees were male).

For the purpose of clarity in the following discussion the interviewees will be referred to as A, B, C, D, E, and F. The host organisations within which they were working at the time of the interview will be referred to as O1, O2, O3 and O4.

6.4.1 Background and Experience of Interviewees

Table 6.1 shows the higher educational qualifications relevant to HCI of the interviewees in the study. Of course such qualifications are not necessary or even sufficient to ensure proficiency in the discipline, but they indicate the orientation of the specialists towards a career in the field of HCI or Human Factors.
Table 6.1

HCI Relevant Higher Educational Qualifications
of HCI Specialists

<table>
<thead>
<tr>
<th>Qualifications</th>
<th>Specialist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Psychology BSc</td>
<td>C, F</td>
</tr>
<tr>
<td>Ergonomics BSc</td>
<td>B</td>
</tr>
<tr>
<td>Psychology BSc &amp; Ergonomics MSc</td>
<td>A, E</td>
</tr>
<tr>
<td>Machine Intelligence Diploma</td>
<td>D</td>
</tr>
</tbody>
</table>

Interviewees B and D appear from table 6.1 to be the least proficient in psychology, however the content of discussions with them revealed that they had considerable experience with psychological, experimental methods, but B described himself as having "just shop floor experience of doing human factors related work".

All of the interviewees, except A, had at least several years experience in their profession, and all but C were still currently employed by their organisation as Human Factors or HCI specialists working on ensuring the usability of UIs (the one exception had recently moved into a more general development group, but described himself as still doing the same things as before).

6.4.2 Current Host Organisations

The four host organisations employing the interviewees at the time of the study were all multi-national ranging from medium size (around 2000 employees in O4) to very large (around 250 000 employees in O3). They were all divided into specialist or national divisions which operated more or less as autonomous business concerns. O3 and O4 each supplied two interviewees, Their business interests were as follows:

O1 Photocopiers, computer hardware, and software
O2 Computer hardware and software
O3 Computer hardware and software
O4 Software
The business roles of the particular divisions within which the interviewees worked were as follows:

- O1 Development of products (largely of programmable photocopiers)
- O2 Development and marketing strategy
- O3 Research centre
- O4 UK division (general software consultancy)

6.4.3 Team Structures

All of the interviewees worked within specialist groups of one sort or another but the structures were variable and this may have reflected or determined the roles and activities of the group members. In essence the interviewees (and other types of specialist) might find themselves working as specialist contributors on a general design project, or as members of a specialist HCI-oriented project. However, the interviews suggested that there is no clear distinction between project types, and hence roles. The variability between team structures might be more representatively described by continuums such as the degree to which work is research oriented, or the extent to which it is market oriented. It is worth noting that some of the respondents to the features analysis questionnaire belonged to two of the organisations described here. It therefore is safe to assume that projects and design teams were of a similar range of sizes.

In general the common structure for organising HCI skills within a company seems to be within a department (referred to variously as a centre, a group, or a team) which specialises in consultancy, development or R&D which contains sub groups (or teams) with particular domains of expertise who all work in a generally similar way. Figure 6.1 is an attempt to represent a general, typical structure for divisions, subdivisions, groups and subgroups. For the sake of argument we may refer to this as a hierarchy of groups. This hierarchy does not represent a particular organisation (it may only be part of an organisation, and an organisation may only include a sub-set of groups). It does not imply communication or working structures either; it is, in fact, more representative of management structures. The composite groups shown are simply indicative of what was reported by interviewees, and are not meant to be comprehensive.

From figure 6.1 it is clear that skills, not projects, tend to form the basis for
organisation, presumably projects are a less stable organising factor. The typical arrangement seemed to mean that there was a certain degree of insularity around groups of specialists in commercial organisations. Certainly the interviewees comments suggested that the situation tended to be one of working closely only with other employees in the same skills group. A particularly illustrative quote on this issue was obtained from F in O4 whilst he was discussing his role in a particular research project:

"We did do a task analysis. We were not permitted to follow that through to actually doing an interface design for the project, really because of the nature of the project being very collaborative, and being done on multi-sites. What’s happened is that people have carved up parts of the project and identified the tools they
have an interest in building, and unfortunately those individuals have done the UI
designs. Now, while they may have been influenced by the task analysis we did,
it’s doubtful in my mind that they’ve been influenced very much."

It would appear that the effectiveness of any work that is carried out by HCI spe-
cialists may be obstructed by organisational separation of specialists in such a way
that there is a physical as well as a disciplinary separation between HCI specialists
and various design specialists.

Figure 6.1 illustrates this constraint which has great influence upon the activities of
HCI specialists. Most of the individuals in computing R&D or specialist consul-
tancy come from a computer science background. Most HCI specialists do not.
More than half of the interviewees mentioned that their group was regarded with
some suspicion or incomprehension by other groups who knew little about their
potential contribution to design. One interviewee summed up this view by saying
"we used to be nicknamed the flower arrangers". Another stated that it was an up
hill struggle trying to convince other company employees that HCI could and
should be considered for all UI design. This finding supports the idea that different
skills groups do not necessarily communicate extensively, and if they are physically
separated then effective communication and collaboration becomes even less likely.

6.4.4 HCI Specialists in Commercial Working Environments

In Stage 1 of the interview, specialists were explicitly asked to describe their roles
within their organisation. However, additional information relating to specialists
roles was also obtained during the other stages. The findings here are drawn from
the entire discussions of the interviewees.

The interviews indicated that there were three important aspects of the specialist’s
job within commercial environments. These are the working regime which in many
ways constrains what the specialist does; the roles themselves which define the aims
and functions of the specialist; and the activities which subsume the various roles.
For each of the organisations employing the interviewees a brief outline of each of
these facets of their job is provided in the following. These summaries are useful
because they provide a great deal of qualitative information about the special nature
of the work of HCI specialists as distinct from systems designers in general. Note
that working regimes are shared by individuals from the same organisation, however roles, and the activities they involve may be different for different HCI specialists within the same company.

**Specialist A (in Organisation O1)**

**Working regimes**

O1 involved fairly well rehearsed and routine activities with the various specialists' roles being well defined. Design projects focused upon updating or producing new versions of existing systems with well defined requirements. Projects were therefore run in a relatively regular fashion, with different groups in the development team (including an HCI group, a UI group, a graphics group, a systems engineering group and a product planners group) collaborating within a well structured design "programme". In brief, the various groups involving the different specialists would collaborate on producing requirements and functional specifications. The HCI group would then deal with presentation and dialogue in the UI and produce a specification which would be prototyped and eventually specified in a form from which software could be generated by programmers. Meetings and informal communication between various groups were used to ensure that the specifications produced by any one group were realistic in terms of feasibility constraints understood by other groups. Repeated user evaluations would be carried out on simulation prototypes in an iterative design cycle, until late evaluations would simply validate that requirements specifications had been satisfied. Although deadlines would be set for completion of the development, A stated that he was able to defer production in the case of serious usability problems.

**Roles**

A's roles were relatively well defined, being limited mainly to contributing to, and working from requirements and functional specifications, and also to ensuring that prototypes met requirements. He kept himself informed on UI issues, to the extent that he discovered a technique for specifying and evaluating UI behaviour (Harel's Statecharts, 1987) which he used to support communication of functional requirements to programmers. He generally worked as a permanent consultant carrying out some design specification and some evaluation work on one or more projects in
Activities

A's activities included reading and keeping abreast of current issues in HCI, attending project meetings, contributing to user requirements and functional analysis, specification of requirements and functionality, dialogue and display design, and a good deal of running experimental user trials on prototypes.

Specialist B (in Organisation O2)

Working regimes

O2 was a large organisation with a small HCI group which meant that the main functions of the interviewee were to provide UI design knowledge for the greatest possible number of people within the company. The size of O2 and variability of the products produced meant that B was required to work on very different projects, ranging from writing design guidelines to very late user evaluations. The company allowed its specialists to take part in research projects. Most notable being one to develop a methodology for technology transfer of HCI, the results of which began to be implemented before the research was complete.

O2 did not have the kind of sophisticated prototyping tools to enable early simulation prototypes to be built and evaluated (as in O1), so results of evaluations tended to feed into later releases of products. In essence there was no single, fixed working regime to which B adhered. The main feature seemed to be his relative autonomy within the bounds of tasks he was given and evaluation of his work by his manager. There did not seem to be any well structured interactions or communication paths between B and other design specialists, his main channels of communication seemed to be through senior managers and junior staff whom he supervised. In other words implicit organisational constraints seemed to be imposing a hierarchical interaction and communication structure as opposed to the more horizontal one in O1.
Roles

B's roles included a considerable amount of research. O2 was involved in some government funded initiatives, the main one concerning B was aimed at developing a methodology for technology transfer, another which B was not personally involved in was aimed at developing a knowledge-based software engineering environment which would support user-oriented design. B also carried out research and design whilst developing user-oriented methods and guidelines for UI designers. The rest of B's time was devoted to consultancy on specific products, usually to solve problems relatively late in the design process.

Activities

B engaged in a wide variety of activities in his different roles. Reading various literature and reports (both on scientific research and on company methods and products) relevant to current projects, conference attendance, correspondence and other activities enabled him to keep abreast of developments in HCI and systems development methods.

Consultancy involved task analysis, requirements and functional UI specifications, evaluation, design solution generation presented as recommendations. Technology transfer involved collaboration on developing and running workshops and writing guidelines. C undertook to set up a user evaluation laboratory which at the time of the interview had been used for some late user evaluations. He also carried out a certain amount of administration and supervised two junior staff members (sandwich course students).

Specialists C and D (in Organisation O3)

Working regimes

The general flavour of the activities in O3 was informal. Communication and control of projects could be both hierarchical or horizontal; "It can be very hierarchical; you can have instructions issued from high above, but we can work horizontally through our network of contacts. We tend to get involved on a given project when a horizontal request comes from another department."
The HCI specialists worked mainly on evaluating specifications, documentation and prototypes, and ran a user evaluation laboratory. However, they were also managing a long term research project to develop a multi-media system for office tasks which was based upon their own ideas (as opposed to someone else's specification). The general regime for evaluating specifications and documentation was fostered by the distance between the groups who produced such material from the HCI specialists' group in O3, and by the electronic mail (email) network between these groups. Specifications and Documentation would be sent in draft form to O3 where the specialists C and D would then read it and email their comments back in an informal manner, or, more formally, send their comments and revisions to a liaison officer who would distribute this information to the relevant groups.

Prototypes were evaluated regularly in the evaluation laboratory which was monitored by C and D or by junior staff. C and D devised their own methods and produced recommendations on the basis of their findings which would be distributed in the same manner as recommendations for specifications and documentation.

At the time of the interview C and D were managing a project based upon their own design concept. The actual development of the research prototype was carried out in a computer science department of a university. Regular meetings and demonstrations of the prototype enabled C and D to monitor progress and update their ideas. The manager of the research group in the university acted as a specialist consultant and contributed to some of the ideas on the project.

Roles

C and D differed only slightly in their roles, which seemed to be very informal, with D taking on slightly more in the way of experimental evaluative work, and C, having moved into a development group, doing more in the way of project management. Essentially their roles were restricted to consultancy and some design. Their consultancy was restricted to informal evaluation of documentation and UI designs, and running experimental evaluations of UIs. They did not continue to educate themselves other than by gaining practical experience. As D stated "We don't base our comments on HCI research findings or guidelines, we use our experience of what has happened in the past, essentially from previous products, all of which were tested on users. They did not carry out research themselves, although they managed
a research project, and they were not engaged in any technology transfer activities such as producing guidelines or running workshops.

Perhaps the most unusual feature about C and D was that they instigated their own long term research project to design and develop a prototype multi-media office system, based upon their own ideas. In this case they could be described as having taken on the role of the main system designers, and in this design project it was the needs of the individual end-user (rather than an organisation or some set of powerful functionality) which drove the entire concept.

Activities

The HCI specialists worked mainly on evaluating specifications, documentation and prototypes, and ran a user evaluation laboratory. However, they were also managing the long term research project.

Activities undertaken by C and D included creative design, informal task and user requirements analysis based upon comprehensive literature on office tasks, informal functional specification of the UI, empirical evaluation and creative solution generation based upon activities in the usability lab, report writing, correspondence, administration and management.

Specialists E and F (in Organisation O4)

Working regimes

The specialists in O4 seemed to have the most flexible working regime which covered a number of activities. The interviewees described the organisational structure as "more of a matrix than a hierarchical structure". There were a variety of skills centres within the company (e.g. communications, and consultancy which the HCI specialists belonged to). On any one commercial project specialists from a number of skills centres would be brought in. However, once a project group had been set up the design team would be very hierarchical with managers controlling the various specialist activities.

Commercial design projects would be "controlled in a very procedural manner"
which was governed by an in-house project control system to which everyone would have to adhere. For research projects such a regime would be rather impractical, since the nature of the work involved would be somewhat unpredictable depending upon the aims. E and F tended to do a good deal of research work as well as commercial consultancy. They also occasionally worked directly for external clients on small scale consultancy projects. For these reasons their working regime tended to be highly variable.

Roles

E and F acted as self educators, reading relevant books and references, attending conferences and so forth, in order to keep abreast of developments in HCI, researchers, UI design tool users and to some extent developers, agents of technology transfer talking to or presenting technical symposia to others in the company, and as consultants on a wide variety of projects. In terms of self education both E and F were probably the most well informed on HCI research and techniques, and E also had considerable familiarity with software development methods having designed a method for mapping CLG specifications. E tended to engage in more managerial activities having been in the company for a greater length of time than F.

Activities

Both E and F worked largely with structured task, requirements and functional specification methods, and development of prototypes. They read widely and attended conferences where they also frequently presented papers. They applied a variety of scientific and HCI techniques, including TAKD (Johnson et al, 1984), CLG (Moran, 1981), and Statecharts (Harel, 1987), for the purposes of task analysis, user requirements analysis, requirements and functional specification. They solved usability problems and engaged in a certain amount of creative design, both of UI prototypes, and of extensions to existing HCI techniques. They ran demonstrations of interfaces, produced regular reports for management, clients and so on, and supervised junior staff.

Unfortunately there was too little space in their company's building to allow a usability lab to be set up. E and F were the only interviewees who did not have access
to this type of facility. This may have been their main reason for concentrating upon theoretically based HCI DETs as a means of evaluating usability, whereas the other consultants tended to use empirical methods.

Summary of Working Regimes

In 03, the two interviewees communicated regularly with employees in divisions in the US, however in general interviewees generally worked closely only with people in their own branch or local divisions. The HCI specialists, and other types of specialist in 04 often worked on projects for external clients on a consultancy basis, however the other interviewees only supplied consultancy within their own organisation. Research projects might involve collaboration with other organisations, but generally organisations would work fairly autonomously on their part of a research project.

Working regimes varied extensively for the different organisations. Table 6.2 gives a simple overview of the differences between the organisations involved in the study in terms of three dimensions or features which were notably variable amongst the different working regimes. The extent to which the organisation is viewed as very or hardly hierarchical, structured or variable, is restricted to comparisons between the four organisations involved in this study. This classification should not, therefore, be seen as conclusive.

Some organisations were notably more hierarchical than others in their working regimes. Hierarchical working regimes such as that of 02 seem to structure communication and control in a top down or bottom up manner with senior personnel acting as coordinators of information channelled to and from junior personnel. In less hierarchical regimes such as that of 01 junior personnel communicate more freely and coordinate their own activities. In 04 there was a mix of hierarchical and horizontal communication and control.

Structured working regimes seem to evolve partly because of the nature of the company’s business. In 01 the work was fairly predictable, being focused around updates of old systems, or new designs based upon well understood application domains. This seemed to enable a routine working programme to be built up. Another factor underlying the apparent structure of working regimes may be the use
Table 6.2
Characterisation of Variation in Working Regimes
of Organisations in the Study

<table>
<thead>
<tr>
<th>Feature</th>
<th>Extent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Very</td>
</tr>
<tr>
<td>Hierarchical</td>
<td>O2</td>
</tr>
<tr>
<td>Structured</td>
<td>O1</td>
</tr>
<tr>
<td>Variable</td>
<td>O4</td>
</tr>
</tbody>
</table>

of scientific methodologies, as was particularly common in O4. On the other hand, in O3 the working regime was highly unstructured and this seemed to be partly due to the use of informal methods of evaluation by C and D, but also to their own preferences and seniority which gave them greater freedom.

Variability of the working regime was not necessarily related to its structuredness. In O4 where work was generally well structured, the variety of project types meant that structures were repeatedly changed to deal with the new product's demands. In O2 the fact that the HCI specialists in the company were so few and the company so large meant that a wide range of working regimes had to be adopted to tackle jobs including experimental UI evaluation and development of workshop based technology transfer courses.

The working regime tended to dictate the manner in which specialists were expected to achieve usability and this varied enormously between organisations. It may be true that specialists have some say in determining their own regimes, particularly if they are senior personnel as in O3. However, it seemed to be more a function of the organisation than of the individual since both of the specialists from O3
shared many working practises as did both of those in O4, despite their differing backgrounds and skills.

A number of activities which subsumed various roles of the interviewees were identified. These are summarised in table 6.3. These activities imply a number of basic actions including such things as reading literature, generating ideas, thinking, writing up research or work achieved, listening and speaking, interviewing various people, designing, running, and evaluating experiments, and so on.

As an exercise in simplification the various activities and implied basic activities are
summarised as four rather gross types referred to as information collection, invention, analysis, and communication. This is not a formal classification, as it is possible to argue that any of the activities above implies more than one of these types. However, the purpose here is to attempt to cover the quality of all of the actions observed, and hopefully it should not be possible to argue that any of the activities above does not fit into any of these classifications.

information collection includes all activities that increase the knowledge of the collector. This may represent reading, listening, observing, empirical data gathering and so on.

Invention includes all creative activities especially design activities where novel features are added in, as far as this analysis goes, a mysterious way to existing knowledge or to the state of affairs. Invention may be dependent upon all of the other activities identified here, however it is not within the scope of this study to demonstrate how.

Analysis includes all activities where information, either collected or based upon invention, is condensed, elaborated or transformed. For the HCI specialists in this study, analysis was usually dependent upon information collection, rather than on invention. Any paucity in the information collected would lead to limitations in the analysis. Analysis would cover, amongst other things, task analysis where interview information might be summarised and classified as it has been in this chapter; application of statistical techniques to raw data providing summaries or clarifying implicit properties of the data; and transformation of informal English descriptions into more formal statements about the properties of some system and its states, and vice versa.

Communication includes all acts which produce representations of information which will permit other agents to collect information. This naturally includes all speech acts, writing, diagram and chart production and so on. It may be that all communication implies analysis of some sort, no matter how basic. However, since the reverse is not necessarily the case, communication is a separate classification.

How important the various activities are really depends upon the importance of the
role of the specialist who engages in them. As indicated by the descriptions of the
various aspects of an HCI specialist's job, including working regimes, roles and
activities, it is clear that different interviewees in the study differed in their roles,
particularly if they were working in a different organisation.

A number of possible overlapping roles largely relevant to user-oriented design (i.e.
not including managerial or administratory roles) fulfilled by HCI specialists were
identified. These include:

*Self Educator:* requiring the specialist to keep up with advances in the field.

*Designer:* requiring the specialist to create a new UI or technique.

*Researcher:* requiring the specialist to undertake work for the advancement of
knowledge in some field.

*Consultant:* requiring the specialist to contribute to a commercial design project.

*Agent of Technology Transfer:* requiring the specialist to teach others about HCI
or pass on skills or techniques.

These roles were not explicit titles used by the specialists, however they are useful
labels which will be adopted here to indicate the rational purpose of the activities
undertaken by the specialists. They are also useful, when thinking about what HCI
specialists do, as implicitly involving certain subsets of the activities identified here.
However, in practice, the activities involved in each of the roles of the specialist
vary within the roles depending upon the circumstances within which the role
exists; for example the HCI specialist as a designer may be designing a system and
its UI (as in O3) or a dialogue (as in O1) or an extension to an existing HCI DET (as
in O4).

In table 6.4 the four types of activity are related to the possible role types of HCI
specialists which the interviews suggested they were involved in. These mappings
are based upon the general descriptions of the HCI specialists and give an indication
of the quality of each of the types of role observed. As indicated, some types of
activity are less important for a given goal than others, and some are not a part of
Perhaps the most important features of table 6.4 to note are that communication activities are probably the most important to HCI specialists, being crucial to most of their roles. It is important to recall that communication is viewed here as requiring some analysis and indirectly some information gathering. This is reflected in table 6.4, however for the role of technology transfer information collection and analysis are represented as less important because they may have been carried out by others or completed by the HCI specialist in the past, and it is the communication which is the main feature of this role.

Also noteworthy is that the roles of researcher and consultant are very similar. The main difference between the roles appears to be in the constraints within which they operate; in O3, where a small amount of research was undertaken by C and D, and O4, where research was more common, the specialists C, D, E and F all agreed that research was much less constrained than commercial work in terms of choice of aims and methods. However, many of the same activities, and presumably skills, come into play within both roles.

A brief qualitative discussion of each of the roles identified, as listed in table 6.4 is included in the following.

*Self education* is a label used in this study to refer to the role of increasing one's own knowledge of a subject area in a deliberate manner (as opposed to gaining experience in an incidental manner, as a by-product of other activities). It was seen to be important, but restricted mainly to reading in most cases, although some practical self education was noted in O1 and O4 where the specialists practised a number of specification and analysis techniques (which became skills) for possible future use. Self education simply involves the specialist keeping informed on HCI, psychology, systems engineering (research and techniques) or whatever he/she considers necessary, and also on general company and project concerns which relate to the work being undertaken.

It is an implicit and private, rather than an explicit role of the HCI specialist.

Only C and D claimed that they did not continue to educate themselves by keeping up to date with developments in HCI and Psychology. They stated that they always
Table 6.4  
Qualities of Activities Involved in  
the Various Roles of HCI Specialists in the Study

<table>
<thead>
<tr>
<th>Roles</th>
<th>Activity types</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Information</td>
</tr>
<tr>
<td>Self Educator</td>
<td>X</td>
</tr>
<tr>
<td>Designer</td>
<td>x</td>
</tr>
<tr>
<td>Researcher</td>
<td>X</td>
</tr>
<tr>
<td>Consultant</td>
<td>X</td>
</tr>
<tr>
<td>Technology Transfer Agent</td>
<td>x</td>
</tr>
</tbody>
</table>

X = A Major Quality  
x = A Minor Quality

based their consultancy or design ideas on personal experience with previous systems.

*Design* is a role name used here to refer to creation of entirely new things or new arrangements of existing things, largely by inventiveness, and analysis of current states of affairs. Perhaps it is best referred to as a poorly understood art form (Maclean 1988). When practised by the HCI specialist, design seemed to be restricted, in most instances, to extensions of HCI techniques for UI design and evaluation, or to guidelines and educational or technology transfer material. The most obvious instance of HCI specialists undertaking system design was in O3 where
specialists C and D were seen to be the main system designers on a long term research project to develop a future multi-media office system, where the focus was on the UI.

For HCI specialists as systems designers, the main source of information used as input to the design, where the specialists were most clearly instigators of a design project, was invention based upon knowledge that the specialists already had. Initially they were short of funding and had to base their understanding of the application domain of office tasks upon reported research by other researchers, and on discussions with management within O3. Having worked out some office tasks and needs which were generally agreed to be representative, they used these to define a "function set" which drove initial prototyping of the product in order to determine how the functionality should be delivered to the user. The main activity providing the information which shaped the early form of the system after this took the form of a number of all day "flushing out" sessions which were essentially creative design and verbal discussion sessions between C and D in which the chosen functionality was integrated into an informal coherent system specification.

How representative this is of HCI specialists as systems designers in general is not clear from this study. However, it is probable that this kind of highly inventive and informal system specification activity on the part of HCI specialists is the exception rather than the norm. In O4 F also undertook a small amount of system design for a WIMPs UI management information system (MIS) to front end a mainframe application. His approach in this case contrasted with that of C and D because he carried out his own task analysis and used much more structured techniques to describe existing users tasks in order to clarify required system functionality.

In most other examples given by interviewees in the study, the original requirements and design concepts for system UI designs were presented to the HCI specialists by other parties. Most of the HCI specialists inventiveness seemed to be devoted to finding ways of ensuring that the requirements and functional specifications supplied by others could be fulfilled by a user friendly system.

Both E and F designed modifications to existing HCI techniques in order to analyse tasks which would have to be supported by UIs of systems supporting a variety of applications. They converted requirements based upon these into system
specifications which met any functional requirements defined by themselves or others. When designing extensions to such techniques their creativity was based upon the needs of the moment "You do what you can in the circumstances; we add things and use different techniques". For example they stated, "TAKD [Task Analysis for Knowledge Descriptions (Johnson et al, 1984)] worked quite well for us, but it did need extensions because it doesn't capture context information very well."

In O2 specialist B undertook to design some guidelines for use of colour on screen, for screen design and keyboard design. These guidelines were for distribution to systems designers in the company. In terms of the proportion of inventiveness versus that of information collection underlying the role of design, this is the most information collection intensive of the design examples observed here. B based his guidelines variously on functional specifications of proposed systems or keyboards, reports on committee findings, reports from company marketeers and someone representing the potential users of the UIs. He also referred to a fair amount of literature and based many of the guidelines on other existing guidelines. In this instance he was acting partly as a self educator at the same time. Since some of his guidelines were borrowed from others, B's role cannot be properly considered design.

Specialist A was responsible for designing the details of dialogues between users and programmable photocopiers, however the degree of freedom was very limited because accepted dialogue styles already existed and were maintained in the new versions which O1 produced. Furthermore the input devices supplied by the machines were often very crude (some were quite sophisticated) and once again this limited what could be designed.

Research is a label used in this study to refer to a role which encompasses activities which increase general (rather than an individual's) knowledge about a subject. Research may be long-term, with no immediate benefits, as it sometimes was for E for example, who managed and contributed to the development of a tool, with government funding, for mapping CLG specifications to JSD specifications, although his company O4 made no use of such tools at the time.

Specialist A concentrated on consultancy and carried out very little research. Specialist B did not engage in a great deal of research, his roles were more restricted to
consultancy and technology transfer.

Specialist B’s organisation (O2) was involved in a number of government funded research initiatives. These were very large-scale projects involving a number of collaborators. The main one involving B was to develop a methodology for technology transfer of UI design methods to non-HCI specialists.

In the main, specialists C and D did not carry out research, their roles instead being mainly consultancy oriented. The multi-media office system design project in O3 was officially a long-term research project rather than a straightforward design project. It was instigated by C and D who acted as the initial system designers. It involved a small amount of research on their part into what constituted a representative set of office tasks. However, most of the research on this project was carried out by a university research group which investigated ways of providing the kind of functions required by the specifications that C and D had designed. Furthermore this group had to provide an environment which supported the philosophy adopted by C and D where their system was thought of in terms of objects which could provide services, have a variety of representations and be independent of any particular machine. Consultancy is seen in this study as referring to the application of knowledge to particular projects, or to solve particular problems, for clients or other groups within the organisation. It is perhaps the core role of the commercial HCI specialist. It certainly represented the main part of all of the interviewees’ jobs. Specialist E described previous consultancy experience with his previous employer in the following manner "... it tended to be bitty; a lot of firefighting [very late evaluation]. You’d be dragged on at the end of a project when a lot of things were going wrong, for example I was dragged onto one project at about four months from its conclusion. They’d had ten people working on it for four years and they hadn’t put any help in; that was fairly typical."

For A, consultancy involved contributing to a number of projects in a well defined design programme. He used system and user requirements generated collaboratively to drive his dialogue specifications which were based upon his experience and the use of a formal state specification notation. As well as developing dialogues, he ran experimental evaluations on prototypes. B undertook a variety of projects as a consultant, he supervised setting up a usability lab, ran experiments on prototypes and analysed user difficulties. He also contributed to the development of a UIMS
by ensuring that low-level incompatibilities between the two tools, which were integrated to form the UIMS, were resolved. He did not develop parts of systems himself, his consultancy role emphasised evaluation.

C and D evaluated a number of different manuals, sets of documentation, and prototype systems. Their role as consultants involved informal evaluative commentary on copies of incomplete designs which were sent to them, to be evaluated experimentally or otherwise, from remote design and development sights. E and F consulted for both external and internal clients’ projects in O4 which makes them the exception in this sense. They were also exceptional in that they worked very largely on development and theoretical analysis for the purpose of evaluation as opposed to empirical, experimental evaluation. This was mainly because of a lack of space for experimental evaluation facilities in their organisation.

Technology transfer is the label applied here to the role of educating others to take over from the consultant by passing on critical information or recommended methods. Unfortunately where technology transfer was seen to take place in this study there was usually little time available to pass on a great deal of knowledge.

Specialist A discovered a specification technique which he thought would be ideal for communicating his ideas in a precise form to programmers. He informally encouraged the programmers to learn the notation and from then on used it to represent his design recommendations in the form of programmable ready material. At the other end of the spectrum of organisational structuredness of technology transfer, B was involved in a large scale research project undertaken to develop a detailed procedure for technology transfer which involved producing training material and preparing workshops, and paper tools to help the receivers of the new technology to apply it.

Somewhere in the middle, in terms of organisational formality, E and F presented technical symposia within O4, as did other types of specialist in the company. This activity was aimed at allowing company employees to “identify who had certain skills.” C and D also regularly talked informally to people in the company with different roles to their own, such as systems analysts and designers. They would "try and persuade them to take a more user-centred approach" and found that, with persistence, they got a very positive response.
C and D did not appear to engage in technology transfer, and did not receive transferred technology from other specialists in the company. It seems likely that O3 itself did not encourage this type of activity. However, O2 and O4 clearly placed great emphasis on this potential role of employees as a valuable educational exercise, although the motives for encouraging it were at odds. In O2 the aim appeared to be to save employing more HCI specialists, and in O4 it appeared to be to ensure that everyone knew what various specialists had to offer and who they were.

Table 6.5 summarises the various roles identified for the HCI specialists in the four organisations studied. The only role not undertaken by all of the interviewees, at least to a minimal degree, is that of technology transfer, which C and D in O3 did not engage in. Self education was seen to be minimal in O3 (restricted to company related reading, and parts of a few books relating to the multi-media office system project). The role of researcher seemed to be limited for A in O1 and for C and D in O3.

Organisations O2 and O4 seemed to involve the most varied roles for their HCI specialists, and this compares with table 6.2 where these two organisations are presented as being the most variable in working regime for the interviewees.

6.4.5 General Summary of Working Regimes, Activities and Roles

Working Regimes

The working regime of HCI specialists appears to vary widely. From the interviews it appeared to do so more as a result of the organisation’s practices and the diversity of projects undertaken, than the preferences of the individual. Since the working regime tends to dictate the roles and activities of specialists, and thus the manner in which they attempt to improve usability of UIs, it would seem probable that the selection of any particular technique for UI design or evaluation will be influenced by factors external to the individual specialist. Namely any factor which predetermines the nature of analyses made possible and the communication channels open to the analyst.
Table 6.5
Roles of HCI Specialists
In the Four Organisations in the Study

<table>
<thead>
<tr>
<th>Roles</th>
<th>Organisation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>O1</td>
</tr>
<tr>
<td>Self Educator</td>
<td>Yes</td>
</tr>
<tr>
<td>Designer</td>
<td>Yes</td>
</tr>
<tr>
<td>Researcher</td>
<td>Minimal</td>
</tr>
<tr>
<td>Consultant</td>
<td>Yes</td>
</tr>
<tr>
<td>Technology Transfer Agent</td>
<td>Informally</td>
</tr>
</tbody>
</table>

For example, if the working regime is very hierarchical, a specialist may not be free to carry out direct task analyses with prospective users in order to determine the appropriateness of design decisions. In O2 which was hierarchical in its working regime, specialist B had little direct access to prospective users for the purpose of ascertaining the utility of design guidelines he was developing. Another example relating to communication is where F used CLG specifications to describe the functionality and dialogue of a UI. The working regime of O4 meant that he was somewhat isolated from other design team members, and that he began working on the specifications at the same time as prototypes were being developed. The result was that, by the time the specifications were complete, the other team members were unwilling to alter their design to conform with them.
Activities and Roles

The wide variety of activities undertaken by HCI specialists represent the diversity of tasks they may be expected to carry out by their organisation. It appears that, for specialists in small HCI groups working for large organisations, HCI involves making a small contribution to a large number of projects. This was particularly apparent in O2 and O3, the two largest organisations. In these cases, the emphasis for HCI activities was strongly on evaluation rather than on design and development.

The roles described above help to clarify the purpose of the activities identified by the interviews.

Self Education is a private role which enables the specialist to keep up with new developments and techniques in the discipline, as well as keeping informed on company matters and the relevant projects to which he or she might be expected to contribute.

Design seems to be highly constrained for HCI specialists in most cases. Limitations in terms of time, and resources, as observed in the features analysis and the designer interview study, operated for HCI specialists also. Frequently their freedom to apply ideas to design was restricted to recommendations or criticisms of existing designs. Where design took place it appeared to be the exception, rather than the norm and tended to be associated with research projects as in O3's multimedia office system research project, and O4's government funded research project to develop a CLG-JSD mapping tool.

Research seems to be a luxury for commercial HCI specialists. In this study, research was observed as being "farmed out" to a university, by C and D, and in O2 and O4, where research took place, it was largely supported by government funding as in the example above. It is interesting to note that research in HCI emphasizes a variety of activities including technology transfer, knowledge based software engineering environments which foster user-friendly design, multi-media office systems, tools for mapping one type of system specification to another. This variety contrasts with the greater emphasis on evaluation in consultancy.
Consultancy seems to be the core role of HCI specialists in commercial environments. There is a distinct emphasis on evaluation, illustrated in particular by O2, O3 and F's previous employer, rather than on design or development in their consultancy. The explanation for this may be rooted in working regimes where communication channels and organisational insularity obstruct the proper integration of HCI specialists early on in design. There may also be a problem with attitudes of computer science educated specialists who have little idea of the skills on offer from HCI specialists, and find it hard to communicate with them. It may also be true that there are not many user-oriented design and evaluative techniques available to HCI specialists which are easily applicable to development; in O4 existing techniques frequently had to be modified.

Technology transfer varies widely from being nonexistent as a role as in O3, through being a recommended company policy for all specialists as in O4, to being a highly structured and organisation-wide activity, as in the research project in O2. It may also take place informally on the initiative of the individual specialist as in O1 and O4. Technology transfer seems crucial for gaining increased acceptance for HCI as an outsider amongst the more computer-centric specialisations in commercial organisations. It was particularly noticeable in O1, and O4 where informal transfer took place, that the results were very encouraging, in O1 an efficient shared notation was used to represent functional UI specifications, and in O4 attitudes to and understanding of HCI were improved by discussions.

The preceding discussions have been restricted to the nature of working regimes, activities and roles experienced by HCI specialists. Hopefully HCI specialist's roles in commercial design have been somewhat illuminated by these findings. This information helps to provide a clear picture of the nature of the working environments within which HCI DETs or less theoretical techniques are applied by HCI specialists to UI design and evaluation, and the nature of the roles which they might have to support.

6.5 HCI Techniques Applied by Specialists in Commercial Environments

In the following sections, the nature of the scientific, user-oriented, design and evaluative techniques exploited by HCI specialists in commercial environments and
uncovered by the interviews will be described in some detail. By scientific I wish to imply anything which is significantly non-heuristic or intuitive. However, this does not deny that a large component of what HCI specialists do may in be, in fact, heuristic and intuitive. It is simply that this analysis does not focus on the features of such activities. The findings should help to clarify what are the features of applicable, scientific techniques for usability analysis in commercial practice.

The techniques which are described in this section are as follows:

- Experimental Evaluations
- Technology Transfer
- Statecharts (Harel 1987)
- TAKD (Johnson et al 1984)
- CLG (Moran 1981)

Unlike the descriptions above the techniques described in the following are not restricted to single organisations, or specialists. Where the same technique, or variations of it, have been identified as used by different individuals in different projects, their experiences with the technique will be presented together and compared and contrasted. For each of the techniques the analysis asks a number of questions which relate to the focus of the study:

* What is the scope of each technique, and how is it used.

* How do HCI specialists using the technique avoid or cope with user-oriented design constraints and problems.

* How does the technique exploit information sources.

* How does the technique support the activities of the HCI specialist, and in what ways does it fail to do so.

Only techniques applied for practical purposes (as well as research) are discussed, those which were mentioned by interviewees as only being used for self education or purely for research were not pursued in the interviews. The techniques identified are assigned a simple classification of their general orientation and purpose as exposed by the interviews. They could be either design and/or evaluation oriented,
and their main function could be *analysis* and/or *communication* of information.

The scoping is intended to suggest the *potential* scope of the various techniques rather than the *scope* of particular applications carried out by the HCI practitioners in this study. The reason for this is that any example would not have demonstrated such scope, and that the specialists themselves tended to modify or only exploit part of the scope of techniques. Another difficulty was that they often used different ways of characterising the scope of their application of a technique. For example the practitioners who carried out experimental evaluations talked about experiments being designed to identify user problems and errors in general, which could have been caused by violation of any or all of the principles. They were not specific about, or particularly concerned about characterising their evaluations in terms only of usability principles and evaluation factors.

The "Maybe" entries in the scoping matrices are used to represent cells which a particular application of the technique may address, but which are not necessarily cells which the technique is explicitly designed to address. Consequently experimental evaluations and technology transfer techniques have a very broad potential scope, whereas the other three more precisely defined techniques have narrower potential scope but more definite areas which they do clearly address.

It is appropriate to add here that the amount of detailed information relating to each technique is dependent upon what was provided by the interviewees in the study who varied in terms of the detail they supplied.

### 6.5.1 Experimental User Evaluation of Prototypes and or Products

*Exploited by:* Specialists A, B, C, and D.

*Orientation:* Evaluation of working prototypes or late evaluation of products.

*Purpose:* Identification of user difficulties.
Scope of the Technique, and How it is Used

Experimental evaluation appears from the interviews to be widely used on commercial products in organisations with HCI specialists. O1, O2, and O3 all had facilities for testing their products on people. Agency supplied individuals, or internal staff and sometimes external representative users were all seen to be used as experimental subjects at various times. The most detailed information on this method came from D who spent a good deal of his time conducting experimental evaluations.

In many cases actual versions of the product are tested, but in O1 there were UI simulation tools which enabled UI testing to be carried out on disposable product simulations of the type described by Rosson et al (1987). The scope of the technique depends largely upon the nature of what is being tested; D stated "We are always testing incomplete products ... Of course it's a fact of life because we couldn't influence the design so much later on." If a simulation of a more complete UI is used early in the design process as in O1, the technique may permit analysts to identify and rectify deep seated problems with a UI. If the technique is applied to an evolutionary prototype (as defined by Rosson et al, 1987) in the late stages of development it may enable analysts to identify major problems, but modifications may be restricted to superficial aspects of the UI. This effect was particularly clear from the descriptions of B who stated that a late evaluation he conducted had no effect upon the first release of the product (which was the evaluated version). "One or two" modifications were made to the second release on the basis of the evaluation, and the rest of the recommendations for modifications were addressed in the third release.

D also noted that findings and recommendations, based upon the results of experimental evaluations, were not generally as influential as they might be; "We are never satisfied with the product when it goes out of the door because we know that there are lots of problems that haven't been fixed. They go out of the door for other reasons, so there is always pressure to get things done [to finalise the design process]. The market, etc may be the deciding factor. I don't have to deal with that side of things; someone else does." Again the insularity of the specialists groups from the other parts of the organisation may mean that it is easier for other groups to ignore usability issues to some extent, and push poor UIs onto the market.
All of the specialists who used experimental evaluation seemed to be familiar with controlled scientific methods, such as used by psychologists, to ensure avoidance of bias, and unwanted variables. In commercial product evaluation typically, test batteries of representative tasks are prepared which are then presented to users in the form of instructions. The users then attempt to carry out the tasks and various aspects of their performance are monitored by the HCI specialists or assistants. D stated that in O3 they restricted their experiments to very simple tasks because the greater the task complexity, the more random noise they found they got in terms of measurements of performance.

Details of this method seem to vary extensively between organisations. The representativeness of users may be very high or quite low, particularly where in-house staff are used as experimental subjects. As D stated, "non-representative users is a problem for us" because in-house staff tended to be familiar with the UI house-style and characteristics of the application.

C and D in O3 developed an interesting experimental method for clarifying users' problems without interfering with what subjects were doing by asking questions. In early experimental UI trials they wished to reveal and analyse problems with the UI, rather than produce representative predictions of end-user performance. They got their subjects to work in pairs; one with a manual telling the other one what to do to achieve the target tasks they had been set. The dialogue between the subjects was usually much more illuminating than having subjects work on their own.

The type of metrics used to judge performance varied depending upon the facilities available in the testing laboratories used by the specialists. Usually time taken to complete tasks, the number and type of errors, and the amount of reference to manuals or on-line help, were recorded for analysis. The experiment might be directly monitored, or more usually video cameras (available in O1, O2 and O3) were used to record the actions of subjects for later detailed analysis. In O1 and O2 the specialists had the facilities to time-stamp and log all keystrokes which helped clarify users' actions if the video tapes were inadequate. Additional, qualitative information was sometimes gained by asking subjects to fill in questionnaires after the experiment, rating what they liked and did not, and describing problems and so on.
### Table 6.6
Usability Scoping Matrix for Experimental Evaluations

<table>
<thead>
<tr>
<th>Evaluation Factors</th>
<th>Usability Principles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simplicity</td>
</tr>
<tr>
<td>Users (Us)</td>
<td>Maybe</td>
</tr>
<tr>
<td>System Application (App)</td>
<td>Maybe</td>
</tr>
<tr>
<td>User Interface (UI)</td>
<td>Yes</td>
</tr>
<tr>
<td>Target Tasks (TIs)</td>
<td>Maybe</td>
</tr>
<tr>
<td>Acceptability of Performance (Acc)</td>
<td>Maybe</td>
</tr>
</tbody>
</table>
The technique may reveal many different types of problem, many of which are difficult to classify. Defining and analysing types of errors, unacceptable task methods, subjective responses, and so on can be very difficult, both C and D remarked that this was a problem for them. Also the value of experimental evaluations appears to be debatable. C stated that experiments were "useless, in that you don’t discover anything that you didn’t know already." He also stated that the use of statistics to reveal "significant" differences in performance was very misleading because such differences can be tiny. However, his colleague was quite positive about the value of such experiments, although he was unhappy that they were sometimes ignored by managers and designers.

Table 6.6 sums up the scope of experimental user evaluations of a UI in the same way as the HCI DETs in Chapter 2 were characterised. Theoretically, an experimental evaluation can be identical in its conditions to the real use of a system. It is clear therefore that there are no limits to the scope of experimental evaluations in terms of usability principle or evaluation factor. With sufficient skill and experience, an experimenter should be able to devise a situation where manipulations of any of the factors (principles or Users, Application, UI, or Target Tasks) would cause changes in performance, enabling the analyst to predict design features’ effects on users in the real world. However, the sensitivity of experimental metrics (such as errors or performance) could be so weak as to force unnatural, and contrived experimental circumstances in order to get any significant differences between the results of different conditions.

Furthermore the accuracy of experimental evaluations depends upon use of representative users, real system behaviour (as opposed to pure UI simulations), and representative target tasks (under realistic conditions). Hence the "Maybe" entries in the appropriate cells. A simple UI simulation could well turn out to be misleading, unless these other factors were accounted for, or their exclusion could be justified on the basis of a plausible hypothesis, for instance, that certain experimental manipulations of colour and contrast on a VDU would affect representative, and non-representative users in exactly the same way.

Avoidance of, or Coping with User-Oriented Design Constraints and Problems.

The main constraints associated with experimental evaluations seem to be the
availability of representative subjects, the timing of the evaluation in relation to the
stage in the product's development, and the extent to which HCI specialists recom-
mendations impact upon the final product. Avoiding the resulting problems seems
to be quite difficult because of the frequent constraints imposed by inadequate or
non-existent prototyping technology, and the time and expense involved in obtaining
representative experimental subjects, analysing data, and making modifications
to the product on the basis of the findings.

If adequate prototyping tools are not available, as was the case in O2 and O3, the
system is usually difficult to modify by the time it is in a robust and complete
enough state to carry out user evaluations. Consequently recommendations based
upon experiments may be ignored, or their impact may be deferred until later
releases of the product as in O2. The only solution to this seems to be ensuring that
prototyping tools and expertise are made available to HCI specialists' groups. Why
this is not already happening seems to be something of a mystery.

Exploitation of Information Sources.

Experimental evaluations have the potential to reveal the precise impact of UI
designs upon users, however the degree to which that information is exploited
depends, at the present time, upon the ingenuity of the analyst. None of the special-
ists reported using any experimental guidelines to enhance the scope of their ana-
lyses, they seemed to rely on their own experience in psychology, or from previous
experiments.

The information sources exploited by experimental evaluations are generally related
to observable performance, e.g. task completion time, number of errors and so on.
This makes piecing together internal confusions a rather difficult task as D stated
"you have to know what the subject's strategy is and you have to be able to decide
whether what they do qualifies as right or wrong; what qualifies as a successful
interaction." The solution used by D, involving getting two subjects to talk to each
other, was able to reveal more about what subjects thought was going on in the sys-
tem because they were forced to explain it to each other. Misconceptions were
made explicit about the nature of the system, its states, and how to accomplish task
goals, based upon information the subjects got from the UI and manuals in the
experiments. Because they were described by subjects to each other they proved
Support for the Activities of the HCI Specialist, and Failure to Provide Support.

Experimental evaluations enable the HCI specialist to test the ability of the UI to successfully support the tasks users are expected to carry out with it. They quickly reveal the nature of problems that end-users are likely to have, enabling the analyst to make accurate predictions. However, they are not so good at explaining such problems because they may not reveal what is going on in the user's head, so the analyst may find it harder to suggest causes and solutions to users' problems. In most cases it may be up to the analyst to carry out a great deal of detective work on the recorded experimental data, and to guess the source of users' difficulties. On the other hand, inventing more revealing experiments such as the two-subject strategy may be a way of improving the value of experiments.

Once the results have been analysed, the HCI specialist has some concrete evidence which should be easy to communicate in order to support any criticisms he or she may have of the UI. Unfortunately, this evidence is often ignored by managers and designers because it is only available after the UI design has largely been completed. As was illustrated by B's and D's experiences, it may only be later releases which benefit from late evaluations. In O1 and O3 late evaluations would also be carried out for the purpose of increasing knowledge about the product's usability, and this knowledge could well be brought to bear upon future products.

6.5.2 Technology Transfer

Exploited by: Specialists A, B, E and F.

Orientation: Design and Evaluation of UIs in general.

Purpose: Communication of basic HCI principles methods and benefits.

Scope of the Technique, and How it is Used

Technology transfer cannot really be described as a single technique, nor is it supported by many techniques at the moment. It does represent an important approach
towards improving the user-orientedness of design. The term here is simply used to refer to any activities undertaken by HCI specialists to communicate HCI information to other non-specialists which will help them to realise the importance of HCI or UI design and evaluative techniques and tools in system development, and possibly help them to apply some of its philosophy and methods themselves.

For specialists A, E and F, technology transfer was an informal activity, or simply involved giving presentations to other employees of their company. This represents technology transfer in its most unstructured form; what is passed on is up to the individual specialist. In the case of A, he passed on some papers to software developers to read, which explained a graphical system state specification notation called Statecharts (Harel 1987). His aim was to use this notation as a means of communicating his UI behavioural specifications in a precise, and unambiguous manner. For E and F, technology transfer consisted of presenting, and attending, technical symposia on research and consultancy projects to audiences of O4's employees. This activity, together with informal conversations with others in the company helped familiarise people with the expertise and any number of techniques (with associated potential value to system development) on offer from the various skills groups in the company.

By contrast, in O2 a highly structured technology transfer project was under way to pass on a basic user-oriented design methodology. The methodology was being developed in collaboration with two other organisations (one commercial and one academic) supported by government funding. The new methodology was not complete at the time of the interview with B. Its main aims and methods had already been decided, however its applicability and value had not yet been established.

B was able to provide a detailed picture of the scope of the transfer activities and the basic user-oriented design methodology. It was essentially designed to fit in with O2's market analysis strategy and seemed to be designed to be highly integrated with existing practice. Assuming that the project to which the methodology was to be applied had already been through the in-house market analysis stage, the methodology involved three phases which were specification of the opportunities for, and feasibility of, the product; identification of the solutions which the product would provide; and finally translation into a definition for the nature of the product itself, including hardware, software, documentation, user-training required, and
<table>
<thead>
<tr>
<th>Evaluation Factors</th>
<th>Usability Principles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simplicity</td>
</tr>
<tr>
<td>Users (Us)</td>
<td>Maybe</td>
</tr>
<tr>
<td>System Application (App)</td>
<td>Maybe</td>
</tr>
<tr>
<td>User Interface (UI)</td>
<td>Maybe</td>
</tr>
<tr>
<td>Target Tasks (TTs)</td>
<td>Maybe</td>
</tr>
<tr>
<td>Acceptability of Performance (Acc)</td>
<td>Maybe</td>
</tr>
</tbody>
</table>
The technology transfer involved running a number of workshops lasting two and a half days, in which syndicates from project teams which had already reached the required stage in their design cycle (i.e. passed the market analysis stage) were required to work through examples of each of the activities that the new methodology required. A syndicate would typically involve a marketeer, two designers, and a technical author, with a user representative being brought in at the end of the workshop to give opinions on the results of the workshop.

The methodology required that the design team consider the various stakeholders in their planned product and describe how it would affect them, and what they might want from it. The general structure of the descriptions would be to work in a top-down manner, beginning with stakeholding groups, then to individual system users (direct or indirect) of the system within those groups, then to tasks, and then objects. The descriptions would be rated according to confidence in their validity, and their importance to the nature of the design itself. Where confidence was low and, or importance high, then the design team members would be advised to carry out investigations to supply the required information to clarify descriptions.

Essentially the methodology was intended to force design teams to be explicit about the impact a system would have on its users, and the roles it would support, and how. One of the requirements of the methodology was that it should be transferrable to non-HCI specialists, and be easy to follow. It focused upon the use of special sheets of paper which the design team were required to fill in, and which helped to guide their descriptions and organise them for later reference. It was simple to follow and did not require specialist knowledge of jargon or specification techniques with all descriptions being written in English, or involving simple charts such as matrices, or hierarchy diagrams.

In the actual workshops, the limited time meant that complete descriptions were not possible, so only one or two examples of each type of description could be followed through with the HCI specialists (e.g. one stakeholder, with one individual user might be described). As B stated, the main intention of a workshop was to "get them [the syndicate] thinking about all the issues, and to do it in a structured fashion."
Table 6.7 suggests that the potential scope of technology transfer techniques in general is without limits. Essentially, whatever anyone is capable of learning may be addressed by a technology transfer technique. The particular technique used in O2 is represented as dealing with user-centred task dynamics in particular, it did not address simplicity, compatibility, observability, consistency and retrievability were not explicitly described by specialist B. However, he did talk about how systems designers were encouraged to consider the impact of their system and the roles it would support. Therefore the tasks that the system would be designed to support would be subject to scrutiny to ensure that they were sensitive to the roles of users and business computer support requirements. Solutions to other areas of usability problems (which the Matrix would capture) seem to have been envisioned as being addressed by designing system training for users. It is also worth noting that the technology transfer technique used by O2 did not address performance effects of the design efforts. It included no explicit methodology for predicting or evaluating user performance, and restricted itself largely to design alone.

In fact the Usability Scoping Matrix is not broad enough to capture some of the issues which were regarded as important in the O2 technology transfer technique. Business concerns and organisational factors are not included in the Matrix because they are not clearly HCI concerns.

Avoidance of, or Coping with User-Oriented Design Constraints and Problems.

The main design constraint for HCI specialists in O2 was that there were too few of them to provide personal consultancy to every design project. The technology transfer effort enabled them to influence a greater number of projects than would have been possible any other way, however the effectiveness of the technique when applied by the non-HCI specialists on the various design teams who used it was not clear, since B had little feedback from its impact.

B described a number of problems associated with the method being transferred. One was that it relied upon the clarity and specificity of the market analysis carried out by the design team prior to its application. If this had not been done properly then the target market would not be a consistent collection of potential clients with similar system requirements which could be satisfied by one product. In one example described by B "the basic problem was that we hadn't done a proper
segmentation of the market beforehand, and so what we got into straight away was this variability problem ... The segmentation was an industry rather than a market segmentation ... but two supermarkets may order goods in a very different way, yet a supermarket may go about ordering goods in a very similar way to that of a DIY store." Unfortunately the method did not include a technique for proper market segmentation. This meant that workshop sessions were sometimes wasted on correcting poor market segmentations, rather than transferring the user-oriented design methodology.

Other problems included the generality of the methodology which meant that it often required considerable modification to suit certain design projects. There was too little time available in the workshops to teach the method properly. The technique itself took a long time to apply and in the workshop, only one or two examples of each type of description would be followed through, "whereas the syndicates have many of each to do and may not bother to finish it because it's difficult after so little practice, and there is so much to do."

In brief, the main value of the technology transfer of a simple user-oriented design methodology seems to be that it increases the number of projects a very small number of HCI specialists can influence. However, the success of the methodology left in the hands of design teams is open to question. B reported that some of the recipients of the new technology had a certain amount of difficulty applying the technique "for example with PC's windowing systems much of this method seems inappropriate where user-groups, users and tasks are not necessarily relevant."

*Exploitation of Information Sources.*

The technology transfer described by B essentially exploits the skills and experience of those who develop what is transferred, and those who decide how to transfer it. Information about HCI and user-oriented design can be presented in a more acceptable form to recipients of the transfer than is generally the case in HCI literature. The methodology described by B was made as non-cryptic as possible, its users were not expected to have expertise in psychology or specialist techniques. Technology transfer on its own simply passes information to those who may require it but to not have the time or the skills to gather information themselves. However, the methodology transferred to the design teams in O2 was itself a means of
encouraging non-HCI specialists to exploit user-oriented information sources which they might otherwise have ignored.

Support for the Activities of the HCI Specialist, and Failure to Provide Support.

The use of technology transfer techniques may not in fact support the role of the HCI specialist in any way. The main value, as mentioned above, is that the specialist can reach more people in less time, however whether this is effective is not known. Two of the other interviewees, E and F, appeared to be very sceptical about the use of user-oriented techniques by non-HCI specialists saying "I would prefer them not to get involved" and "If somebody else has to do an HCI design, all we can do is to give them a checklist and give them guidelines. We prefer not to give guidelines, but if we are forced to do so then we say use it like a checklist. They're things for you to think about, they are not god given facts that tell you to do it this way, because every situation is different. It's about design, it's not about following these steps to a good design." These comments strongly support B's criticisms of the transferred methodology, in that it was too general and needed adaptation for different products, and that there was no guarantee that design teams would use it properly.

6.5.3 Reactive System State and Dynamic Behaviour Modelling
(Statecharts; Harel, 1987)

Exploited by: Specialists A and F.

Orientation: A Graphical Specification Technique for Use in Design and Evaluation of UIs.


Scope of the Technique, and How it is Used

Harel's Statecharts are a simple graphical notation for specifying clearly, and unambiguously the state behaviour of a reactive system (i.e. one that is largely event driven, as are most UIs). They are basically an extended version of state transition diagrams (see Chapter 2; on CCT's user of STN's). In other words, using this
notation it should be possible to represent the result of any action at the UI in any possible state. Statecharts have all of the formal properties of state transition networks (STN's) together with a number of advantages (Harel, 1987):

- They exploit and enhance the graphical strengths of STNs
- They can deal elegantly with notions of hierarchy.
- They capture concurrency (it is possible to be in two states simultaneously where simultaneous events can occur) whereas STNs are essentially sequential.
- They express broadcast communication with a simple mechanism.

Figure 6.2
A Very Simple Statechart

A very simple example of a Statechart is shown in figure 6.2. The result of Action A, (given condition [c] is true at the time, in substate 1.1) causes the system to leave substate 1.1 and superstate 1, and move into superstate 2, and the default substate 2.1 to which the small arrow points. In substate 2.1 actions B, C, and D are valid. This might represent any semantics which the analyst chooses, for example substate 1.1 might be a SELECT-TEXT mode in a TEXT-EDIT state represented by
superstate 1. Action A could be pressing ESC, given the condition that no other command is active at the time, which would take the user out of SELECT-TEXT mode and TEXT-EDIT state and put him or her into a BATCH-COMMAND substate(2.1), with three possible commands; B, C and D, which was the default substate of an INTERRUPT superstate (2).

Specialists A and F used the Statecharts notation extensively, although they both stated that it is not an explicitly user-oriented technique. It was clear from the interviews that Statecharts fulfilled a number of useful functions for that analysts, since they were both very positive about its value. This attitude was significant in the light of the fact that they used the notation without any support tools which could have made their job a great deal easier.

Statecharts was generally used to describe the virtual behaviour of the system, in terms of expected actions and dynamic responses, such as a change in state, or an output. The behavioural characteristics of the virtual system were specified by A and F before any software was actually written, and the specifications were analysed and evaluated by the specialists to make sure they were correct before being passed on to programmers who could write code directly from the specifications.

There were a number of differences between the structure of the projects which A worked upon and that of the project in which F used Statecharts. A used Statecharts on a number of projects to redesign or update sophisticated office photocopying technology and had to coordinate his activities with those of a number of other specialists on design projects, whereas F worked largely alone during the stage of his project where he wrote the system specifications. A used Statecharts to represent the virtual states of the underlying system and to direct a separate, "orthogonal" specification of the precise nature of the interaction which was a dialogue specification which dealt with communication between the user and the system. A had to ensure that these specifications mapped onto one other orthogonal specification of the system which was designed using a UI simulation prototyping tool which captured the "look and feel" of the presentational aspects of UI being designed, but which was not capable of simulating the underlying system behaviour (Trillium; Henderson, 1986). All of these specifications would be built up in parallel, but the Statecharts would tend, logically to precede the other two. Only after these specifications had been completed would the code be generated based upon
<table>
<thead>
<tr>
<th>Evaluation Factors</th>
<th>Usability Principles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simplicity</td>
</tr>
<tr>
<td></td>
<td>Compatibility</td>
</tr>
<tr>
<td></td>
<td>Observability</td>
</tr>
<tr>
<td></td>
<td>Consistency</td>
</tr>
<tr>
<td></td>
<td>Retrievability</td>
</tr>
<tr>
<td>Users (Us)</td>
<td>Yes</td>
</tr>
<tr>
<td>System Application (App)</td>
<td>Yes</td>
</tr>
<tr>
<td>User Interface (UI)</td>
<td>Yes</td>
</tr>
<tr>
<td>Target Tasks (T)</td>
<td>Yes</td>
</tr>
<tr>
<td>Acceptability of Performance (Acc)</td>
<td>Yes</td>
</tr>
</tbody>
</table>
them.

F used Statecharts on a project to develop his MIS in tandem with a task analysis (TA) methodology (TAKD; Johnson et al, 1984). The system was developed "from scratch" using a strongly top-down specification approach. The TA methodology was used to analyse and represent task related information from an initial interview based study of prospective user activities and requirements for system support. The TA methodology, though capable of representing the roles of the system within the task domain, could not represent that behaviour of the UI. As far as B knew "Statecharts are ... the only kind of formalism which can capture the very interactive type of system you're getting now. Also it appears that they are a very good way of communicating with the programming." So he chose Statecharts to model the expected user input and dynamic behaviour of a graphical UI for his own analysis, and to communicate precisely the intended system behaviour to a programmer.

An implicit requirement of Statecharts is that the analyst already has a good idea of what the system scope will be, how it should behave, and what if any exist, are the limits of the system on which the application is to be implemented. Otherwise he or she is in danger of producing an inappropriate, redundant specification that cannot be implemented.

In spite of the difference between A’s and F’s projects and the applications being designed, their use of the Statecharts notation was only slightly dissimilar, and this is probably because the scope of the notation is relatively limited to the simple state behaviour of the system; it says little or nothing about presentation and semantics of that behaviour. However, what the notation lacks in scope, it seems to gain in generality across applications, and in its power of precision of expression.

Table 6.8 shows that Statecharts cannot address user-oriented issues directly, it can only support the representation of application, or UI properties from which an analyst would be able to deduce that a design was more or less usable. It says nothing about user properties, and how compatible or user-centred the design is, it merely represents, or more precisely is sensitive to, simplicity, observability, consistency, and retrievability. Changes in these ideal system properties would have effects on a Statechart representation. Changes in compatibility and user centred task dynamics would probably have no noticeable effect. Likewise, whatever the
users or the target tasks turn out to be, Statecharts cannot suggest their implications for usability. Furthermore Statecharts has nothing to say about performance aspects of systems with respect to users.

Specialists A and F confirmed that Statecharts does not explicitly support user-oriented design, and that it would be quite easy to specify an unusable system using the technique. They had to be conscious of implications for usability whilst making their specifications, and they used other user-oriented techniques to support these other aspects of their approach.

Avoidance of, or Coping with User-Oriented Design Constraints and Problems.

One of the main problems which became apparent with Statecharts in the interviews is its lack of ability to deal with the constraints of the intended application software. It was perfectly possible for F to specify states which would be impossible with the intended software. A got around this problem by being familiar with previous similar versions of the intended product and communications with other groups working on the design. For F this was not so easy because the MIS was a completely new product, without precedent in his organisation, and the application software had not been decided upon at the time when he was writing the specifications.

Another problem which both A and F remarked upon was that the notation did not discourage them from specifying an unusable system. This problem relates to the fact that Statecharts does not deal with presentation or semantics. They had to rely on their own skill and intuition, and other specifications such as the simulation prototyping tool used by A, and TAKD used by F, to deal with this.

Statecharts may however help to encourage the analyst to maintain consistency, as F stated "It shows consistency, or the lack of it, as long as you are bearing it in mind throughout." But the main value it seems to have is its concise representation of the precise nature of the system behaviour which is easy to understand, and captures the way in which context influences the effects of actions: "You can get a conceptual model of the system and know exactly what’s wrong before a single piece of code’s written. I have discovered quite a few mistakes that I myself made ... and because it’s a visual medium, you can process it more quickly than you can digits which is a problem with others [other notations]." As far as A and F were
concerned, Statecharts made the analysis of a potentially confusing and dynamic state of affairs a great deal simpler, and enabled them to communicate much more effectively with programmers who would implement their ideas.

*Exploitation of Information Sources.*

Statecharts is a fairly powerful expressive technique which enables the analyst to transform informal expressions about a planned UI design into formal, visually concise expressions which enhance his or her ability to see the dynamic implications of the specification, without having to resort to writing and running the software. The formal expressiveness should also permit the analyst to prove the validity of any claims made about the effect of various actions given certain conditions in the system.

The Statecharts notation does not assist in the collection of information as does experimental evaluation of a UI, it merely enhances the detectability and precision of information. It is also an easily processed form of representation which can be used to communicate information from one agent, such as an HCI specialist, to another, such as a programmer. Its precision, which is sufficient to make it machine readable, permits a non-programmer to design programs for others to implement.

*Support for the Activities of the HCI Specialist, and Failure to Provide Support.*

The Statecharts notation fulfilled a number of valuable user-oriented design functions for A and F which were not provided by other methods available to them. It supported them in their roles as designers and as consultants. The activities it facilitated were essentially those of analysis and communication of information about the intended behaviour of the virtual machine or system UI. For F in particular, the nature of Statecharts was well suited to a top-down approach to design because it is well able to support "information hiding" which enables details of the design to be specified in more detail at a later stage in the design cycle so that decisions can be put off. This makes the technique quite flexible, which means that it should fit into a variety of design approaches; e.g. top-down as in SSADM (Downs et al 1988), bottom up as in the reuse of existing libraries of packets of code (Sammet 1986, Jacobsen 1987), and middle-out as in JSD (Jackson 1983). A drawback with Statecharts is that although it should be quite easy for most people, including potential
system users, to read them, they do not convey the look and feel of the system. However, in this respect, as abstract system representations, they are not unusual.

Its main functions included providing a clear system specification which could be analysed and discussed before software was written which would be more difficult to modify. It enabled A and F to identify conceptual and logical mistakes they had made in their specifications. It was a concise and unambiguous means of communicating designs to programmers.

At the beginning of this sub-section, a number of advantages of Statecharts were listed. It is appropriate here to expand upon the first of these as it is a particularly important feature with respect to HCI. Harel (1987) claims that Statecharts capitalise on the visual strengths of State Transition Networks (e.g. Arbib, 1969; Parnas, 1969) which are a form I representation technique using graphical constructs to show the states and transitions possible in some system.

Aside from its formal advantages, Statecharts seems an appealing notation because of its graphical character which specialists A and F claimed made it very easy to scan briefly or follow in detail. This graphical quality appears to exploit human visual processing strengths which make pictorial representations superior to text for some tasks (e.g. Snodgrass et al. 1978).

Gestalt psychologists discovered laws which describe how forms are perceptually organised (Hochberg 1964; Kaufman 1974). For example a series of marks "- - - -" may be seen as forming a line rather than as discrete elements. Diagrams such as Statecharts, which exploit these laws, seem to exploit human ability to detect relationships, changes and groupings in information (Kosslyn 1989). This quality may make Statecharts particularly useful for certain kinds of communication or analysis.

The main failures of Statecharts related to the complexity of descriptions required for real systems. As with many descriptive notations, including those described in Chapter 2, Statecharts are demonstrated by Harel (1987) on very simple systems. Non trivial systems such as those being developed by A and F generate far more complex charts and with the complexity come a number of problems, not the least being the physical size of the specifications "You may have an enormous diagram that fills a wall, in such a case, tracing the effects of a change may be
extraordinarily painstaking and difficult to do." By hand it may be impossible to complete a Statechart specification of a complex UI within a reasonable amount of time. Neither A nor F had the support tools which are available to make the job easier. F was using a drawing tool to save time, but he had almost run out of system memory, and still hadn’t completed his specification.

E was involved in the design of a system with a sophisticated, adaptive UI which involved the use of a rapid prototyping tool based upon state transition networks (RAPID developed by Wasserman et al, 1985; see Chapter 3). With this tool he stated that it is "very easy to knock off small prototypes when there are 20 to 30 screens. But given that we had an adaptive interface, at 6 different levels, in 3 different dimensions, which is already 18, and you had something like a hundred screens, then you are talking about 1 800 state transition networks." This kind of problem could also be a problem for Statecharts, even with automated support, however the two systems described by A and F were not nearly this complex and their specifications were correspondingly simpler.

Unfortunately the tools "Statemat" and "Statemaster" (Ad-Cad, 1986 and Wellner, 1989) which are capable of checking for mistakes, translation into skeleton code for system software, and supporting UI design, are highly expensive. At the time of the interviews A was attempting to get a budget to purchase Statemate, whereas F said that his company would not permit such a large expenditure. Without the support tools, error checking and modifications (which sometimes propagated throughout the system, for example when the broadcast mechanism is operating) were extremely time consuming activities. However, both A and F preferred to modify their specifications than allow code to be written and then modified, they were also extremely positive about its value "... even by hand, it’s the best method I know for representing interfaces."

6.5.4 Task Analysis for Knowledge based Descriptions
(TAKD; Johnson et al, 1984)

Exploited by: Specialists E and F.

Orientation: Design of a system’s functionality based upon users’ task knowledge.
Purpose: Analysis of tasks to generate a user's knowledge structure.

Scope of the Technique, and How it is Used

Task Analysis for Knowledge based Descriptions (Johnson et al 1984) specifies a method for analysing information, collected from a wide variety of sources, about people's tasks in order to identify the underlying task executor's knowledge in terms of actions and objects. Actions and objects are classified into generic actions and objects and assigned appropriately to lists of action/object pairs which are expressed in terms of a knowledge representation grammar (KRG).

A brief example follows. Three steps in a hot drink making task description might be:

Boil water in electric kettle.
Place teabag in cup.
Pour water into cup.

A dictionary of relevant generic actions and objects would be:

Actions
    PREPARE: (boil, warm)
    PUT: (place, pour)

Objects
    LIQUID: (water, milk)
    KETTLE: (electric, stove)
    INGREDIENT: (teabag, coffee, chocolate)
    RECEPTACLE: (cup, mug)

Two KRG expressions generated from the above might be:

PREPARE/ a LIQUID/ in a KETTLE
PUT/ an INGREDIENT/ in a RECEPTACLE

This structured technique of describing task knowledge enhances the detectability of similarities between tasks, and it encourages the analyst to be explicit about what
is involved in completing a task, and attempts to cut down on redundancy of description. There is an obvious risk that some of the richness of the semantics and contextual information supplied about tasks will be lost during the analysis (for example, when preparing the TAKD sample above, I noticed that PREPARE might mean chill, which could be done in a fridge. However, a fridge and a kettle are two very different objects, and to class both purely as preparatory devices would be to lose a great deal of task related information about them; for example a reader unfamiliar with kettles and fridges might not know that you cannot simply pour liquid into a fridge in order to prepare it). This may be the price for the condensation of knowledge which this methodology permits.

Only the analysts in O4 used this technique which has much in common with the competence grammars amongst the HCI DETs (i.e. the Formal Grammar of Reisner 1981; and TAG, Payne & Green 1984) described in Chapter 2 (in fact they had explored many more HCI DETs than any of the other interviewees). E used the technique on a collaborative research project which aimed to develop a set of workstation based tools to assist systems analysts in the use of JSD (Jackson’s System Development methodology; Jackson, 1983). F used it to describe the tasks of retail managers, in the design of the WIMPs UI Management Information System to which he also applied Statecharts; this was a commercial project. E was working on a large multi-site project, but F was working largely alone. In both cases the specialists were unfamiliar with the application domain, and there were no comparable systems currently available at the time of the projects. Both E and F used interviews to gather the information they required to develop the TAKD descriptions.

Notably, both E and F found that they needed to extend the notation to capture the task related information they wanted; “TAKD worked quite well for us, but it did need extensions because it doesn’t really capture context information very well. It didn’t help us very much in allocating information between system and user. Sequencing was very difficult to capture.” E added task hierarchies to show relations among tasks and sequences of related or dependent tasks. Both E and F also added ways of including contextual information which enriched the descriptions, and scoping diagrams which clarified user’s tasks which the system must support and those which it might support, as opposed to tasks which required little or no support.
<table>
<thead>
<tr>
<th>Evaluation Factors</th>
<th>Usability Principles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simplicity</td>
</tr>
<tr>
<td>Users (Us)</td>
<td>Yes</td>
</tr>
<tr>
<td>System Application (App)</td>
<td></td>
</tr>
<tr>
<td>User Interface (UI)</td>
<td>Yes</td>
</tr>
<tr>
<td>Target Tasks (TIs)</td>
<td>Maybe</td>
</tr>
<tr>
<td>Acceptability</td>
<td></td>
</tr>
<tr>
<td></td>
<td>of Performance (Acc)</td>
</tr>
</tbody>
</table>
Whereas E sent his TAKD descriptions to other members of the design team, and was doubtful as to how much they influenced the design, F followed through his own extended TAKD specifications by translating the functional requirements for the system they generated into Statechart specifications of that functionality; "TAKD is good for describing tasks in a purely abstract way, but when you want to get on to UI descriptions then Statecharts is far more useful." F found that the TAKD specifications mapped well onto the Statecharts. "In practice it [TAKD] was reasonably well integrated [with the rest of the design process]. That's partly due to the fact that I'm only prototyping the UI, not much functionality."

The most important point to mention, regarding the scope of TAKD, is that it complements the scope of Statecharts which it was used in tandem with by specialist F. As shown in table 6.9, where Statecharts is weak (notably with respect to the User evaluation factor, and with respect to the Compatibility and UCTDs principles) TAKD is strongest. This example of use of complementary techniques provides strong support for the notion that the limited scope of a technique may be a major drawback if it is time consuming to apply and difficult to master. The analyst may be forced, as was F, to learn another technique and spend time applying it in order to cover the possibly wide variety of aspects of the system which are important in a real world design situation.

TAKD focuses on users' existing knowledge of the real world in terms of the familiar objects and actions, and the commonalities which allow generalisation of task rules from one specific instance to another. It is less strong in terms of compatibility than in terms of UCTDs because the process of generification abstracts away from the familiar instances to possibly unfamiliar generalised concepts which may be presented in the UI in such a way that users no longer recognise them. For instance, RECEPTACLE may not easily suggest a mug, more than it does a kettle, and confusions could arise. Certainly, systems designers in O4 had difficulty with the TAKD specifications because of the degree of abstraction and loss of contextual information. They were not sure how to implement and represent generic concepts such that users would be able to understand them.

Simplicity of target tasks may be captured by TAKD representations by counting the rules required to describe all target tasks with the UI but it is not clear that this would be a psychologically real metric, if it is based upon idealised generalisations
of knowledge without additional devices such as TAG has (Payne & Green 1986) to capture properties of users' existing knowledge as distinct from other knowledge.

Since TAKD includes no independent system- or UI-behaviour modelling component it is unable to address simplicity thoroughly, and does not touch upon observability, consistency, and retrievability. It does not suggest performance predictions, since it is strongly focused on design rather than evaluation.

**Avoidance of, or Coping with User-Oriented Design Constraints and Problems.**

Contrary to the recommendations of Johnson et al (1984) it was not possible for E and F to exploit a wide range of information sources, although both wished they had been able to. Organisational and time constraints limited E to eight three hour interviews with systems analysts, and F to several interviews with one highly experienced, representative prospective user. Both of these specialists complained that the TAKD methodology did not provide sufficient guidance for the actual information gathering which it depends upon; "TAKD, like CLG, never tells you how to do task analysis, all it does is suggest or expect you to do interview or observation. There is little in the literature on this. You have to read literature on knowledge elicitation which tells you far more about how to do task analysis than anything which goes under task analysis."

It was obvious from the descriptions of E and F that the TAKD specifications had to be used as communicable representations, of target tasks for the system to support, to prospective users or other design team members. F stated "I did go through the output of TAKD with the person I was interviewing [the prospective user representative whose interviews provided the basis for the KRG sentences]. He had no problems validating it." However, E said "... the output was really for the project team and we did have to do a lot of diagramming and simplification to get points across for comprehensibility to others. The KRG sentences on their own don't mean a lot." It would appear that to those unfamiliar with the application domain, such as members of a system design team, may have considerable difficulty in understanding TAKD descriptions "... it depends on the audience." This may be because of the lack of semantic and contextual richness in the KRG expressions, as noted earlier.
E was worried that the length of the specifications generated from eight 3 hour interviews (three 50 page specification versions). He considered that the mere size of the documents would discourage others from using his descriptions. He also noted that much of the individuality expressed by interviewees' task methods was lost in the KRG specifications. What tended to happen was that variability would be included as options in the KRG expressions making them more structurally complex than they should have been. This problem is precisely the same as the one identified by Payne and Green (1986) who discuss the trade off between competence (ideal), and acceptance (all sufficient) grammars as ways of describing tasks. They selected competence grammars, but where there is no clear optimum method for accomplishing a task, or it depends upon context (which the TAKD grammar fails to capture), then it is difficult to avoid a more acceptance oriented, and hence more complex, task description.

Exploitation of Information Sources.

As noted above time constraints tend to limit the extent of realistic task information gathering exercises. "For the TAKD work we decided all we could do were interviews. We couldn't do observational work since that would have taken far too long because we were looking at a very long procedure; system analysts' work." On the other hand, if used in systems development TAKD encourages the analyst to investigate, as far as possible, the nature of tasks users are likely to carry out with the UI. It tends to force its applier to look carefully at the structure of tasks, and to identify similarities between actions and objects, which could mean that more attention will be devoted to the users tasks than might otherwise be the case; "What we found was that just having a structure to follow for picking out tasks and then laying them down was very helpful."

As far as communication of information goes the technique seems to fail if the receiver of the specifications is not familiar with the task domain from which they are drawn. This seems to be because TAKD, whilst providing analysis, may tend to lose contextual and semantic information, which tends to be picked up in interviews or observations. By adding extensions to the TAKD notation, E and F found that they were able to include more of this information. Even with the extensions non-application domain experts (i.e. the software writers in E's project) were still unable to understand the specifications.
Support for the Activities of the HCI Specialist, and Failure to Provide Support.

O4 did not have an experimental evaluation laboratory, unlike the other organisations in this study. This seemed to mean that the activities of E and F were more restricted to ensuring user-orientedness in the development of systems. TAKD is a useful means of recording and organising the information required to ensure that the intended system supports the tasks that users will want to carry out with it. It may help to increase the chances of getting things right first time, when prototyping with iterative user evaluations is not possible.

On the other hand, since TAKD relies upon analysis by hand, it may be time consuming, but when compared with, for example CLG, it is relatively lightweight "... the notation is not overly heavy; it's quite simple and it does help you to come to an understanding on something that you can then question, and question other people about wanted to do in the project we were undertaking."

Unfortunately, as has been pointed out the technique does not provide much support for the information gathering process prior to the analysis. This may not intuitively seem to be a problem, however if HCI specialists can criticise its lack of support in this area, then the problem is likely to be quite significant for non-HCI specialists. It should be pointed out that more recently Johnson and Johnson (in press), recognising this problem have produced a technique for carrying out such information analyses called Knowledge Analysis Technique (KAT). However, this technique was not known to the specialists in O4.

The process of "generification" is heavily dependent upon the point of view of the analyst, and in an unfamiliar domain unlike the hot drink making one illustrated above (as was the case with E and F's projects), it may be easy to miss, or to make erroneous assumptions about, similarities between actions and objects, unless the analysis is pursued very carefully, i.e. by gathering as much information as possible, from as many sources as possible, and by checking the KRG sentences with application domain experts, which F did.

A possible benefit of TAKD which was not explicitly noted by E and F is its ability to encourage reuse of concepts in the design, which could save on the amount of time taken specifying functionality, and writing code. Objects and actions are
identified and classified as generic types. By identifying a generic type and noting all of its task roles, the analysis paves the way for a complete specification of all of the behavioural requirements of system actions and objects (packages of software, or data structures and procedures) which are specified to replace those in the original task domain.

6.5.5 Command Language Grammar
(Moran 1981)

Exploited by: Specialist E.

Orientation: Design of user and task oriented dialogues for UIs.

Purpose: Analysis from tasks through to details of dialogue and communication of interactive system behaviour to programmers.

Scope of the Technique, and How it is Used

For a full description of Command Language Grammer (CLG; Moran, 1981) the reader is referred to Chapter 2. CLG is the only HCI DET reviewed which was seen to be used by HCI specialists in applied commercial design practice. It is basically intended to be a top-down stepwise refinement specification methodology for the grammar of all possible valid interaction languages for a given system UI, whilst preserving a users conceptual model for the system based upon the structure of the tasks which the system will support.

E used CLG, on a large, collaborative, commercial system design project, where his job was to write a specification of the interactive dialogue for an adaptive UI for an existing email system. The prospective users would be anyone currently using the existing email system. He followed the descriptions in the paper on CLG by Moran (1981) which contained examples of a specification of a simple electronic mailing system. He stated that he was probably influenced by the similarity of the two applications which may have been an advantage, but "it may have led me into doing some things in certain ways which I shouldn't have done." The whole specification took about four weeks and generated 60 pages of notation which were passed onto a secretary to be typed up, and which came back with many typing errors (the
secretary had great difficulty with the notation. E then had to spend a fair amount of time proofreading and correcting the errors.

E found that CLG worked very well with the system architecture which was maintained in the design project. The design team strove to maintain a high degree of separation between the dialogue control component of the architecture, and the application part of the architecture (the original email system software) which communicated with the dialogue controller through an "application expert" which was able to map from the user's interactive tasks to the application software. E used his CLG specifications as a basis for his communication with the application expert programmer. He was able to describe the tasks of the user at the interaction level which was compatible with the level of description used by the programmer. Using CLG he was able to specify the structure and parameters input and also the feedback parameters required of the application software via the application expert.

In spite of this, by the time E had completed the specifications and handed them over to the dialogue software designers they told him (in his words) "We can't understand a word of this specification without spending time we haven't got to understand it, and we've already started doing it this way. Tough luck!" He stated that a lot of his efforts were wasted, however it was a very educational experience for him.

E had also been involved in a research project which aimed to provide a tool for mapping CLG specifications onto JSD specifications. This project was based upon the belief that it is best to allow specialists from different areas to write their own specifications, but that their work could be integrated by provision of a tool which would allow one specification to be translated into another. E noted that the entities in CLG and JSD are quite similar, and that the actions and functions of JSD specifications map quite well onto the syntactic methods in CLG. These similarities were sufficient to permit semi-automated mappings from one specification to another.

E was never able to look at the mechanisms in CLG for representing specifications of commands at the interaction level in terms of rules which can be used to capture similarities between expressions in the grammar (see Chapter 2 for discussions of the CLG interaction level). However, on the basis of his understanding of this
Table 6.10
Usability Scoping Matrix for CLG

<table>
<thead>
<tr>
<th>Evaluation Factors</th>
<th>Usability Principles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simplicity</td>
</tr>
<tr>
<td>Users (Us)</td>
<td></td>
</tr>
<tr>
<td>System Application (App)</td>
<td></td>
</tr>
<tr>
<td>User Interface (UI)</td>
<td>Yes</td>
</tr>
<tr>
<td>Target Tasks (TTs)</td>
<td>Yes</td>
</tr>
<tr>
<td>Acceptability of Performance (Acc)</td>
<td>Maybe</td>
</tr>
</tbody>
</table>
feature of CLG he was convinced that, by exploiting it, he would have been able to model the complexity of dialogue specifications in CLG.

The CLG Scoping Matrix in table 6.10 is reproduced from Chapter 2 where the main justifications for its characterisation are discussed. Suffice it to say that the technique, although time consuming, and detailed in its specification, does not have the extensive scope, with respect to design and evaluation for usability, which its complexity might at first suggest. In the example of its use observed in O4, the system properties which CLG may be capable of representing (the "Maybe" cells in table 6.10) were represented by the analyst, which supported him in communicating with an expert application programmer more successfully. However, this communication did not seem to be supported by the specification when it was passed on to the system implementors who could not understand the notation and did not incorporate it into the design.

The possible performance prediction extensions, and complexity metrics for CLG were not exploited by E, which is not surprising since these are not explicitly supported by the descriptions of use of the CLG approach provided by Moran (1981).

Avoidance of, or Coping with User-Oriented Design Constraints and Problems.

There were a number of difficulties with the CLG notation which made it unlikely that CLG could be routinely applied given typical design constraints, such as lack of time, complexity of the application to be specified and lack of experience with HCI. E, even as an HCI specialist, found CLG very difficult to learn "... some of the aspects of the specification, particularly at the interaction level, are horrendously difficult to understand with respect to what you're expected to do, for example, generating the rules for how you produce command parameters, the notation is very complex and repetitive down through the levels." He felt the specifications were unnecessarily large and difficult to manage. He also experienced problems with the incomprehensibility of CLG specifications when passed onto other members of the design team. Essentially non-HCI specialists had no experience of the technique, and worse still, they did not seem to see its potential value.

There were several important constraints on the commercial project for the design of the adaptive email system UI which severely curtailed E's activities. The first
was that he did not have access to the functional specifications of the existing email systems which meant that he had to spend a certain amount of time "playing" with the system in order to determine its behavioural characteristics. The second constraint was that he "wasn't given the opportunity to look at how people use, or what tasks people do with electronic mail systems in earnest." He had to use literature and his own experiences with email systems. The third constraint was the limited amount of time available to E in which he had to learn how to use CLG and write a correct specification. The fourth constraint was that the software for the system was being written in parallel with his specification activities which meant that the system design was already becoming fixed according to other software engineering ("computer-centric") influences. The research project for the mapping tool was not subject to these kinds of constraint.

E made a number of criticisms of CLG on the basis of other problems in commercial practice. "... with a real system it turns out to be immensely time consuming, and if you make a change at the interaction level, for example, you have to make changes at all the preceding levels and you just can't cope with the immensity of the specification.

So for anybody to use CLG seriously, you'd need tools ... what you really need is something which can take the specification and automatically give you a state transition network, for example, which can then be turned into a first pass prototype ... You build in errors as you go along. Also deciding what goes at what level can be difficult for some people."

*Exploitation of Information Sources.*

CLG requires that the analyst investigates the nature of the tasks which the system is to support, although it is not explicit on how to go about the collection of such information. The main value of CLG, in terms of exploitation of information sources, seems to be that it maintains the content of task analyses throughout a top-down specification of the UI in such a way that knowledge which its users are likely to lack will not be unnecessarily added to the required user's conceptual model for the system; "Whilst the problem of making incorrect design decisions on the basis of too little information early in the design project does exist, CLG's starting point is the right one in terms of users' tasks. If you started bottom-up, you wouldn't be sure of what you would provide except in terms of system constraints on what you
can do."

As stated above, E did not exploit the potential performance metrics suggested by Moran however he did identify potential user error sites in the specification by investigating mismappings between the various levels of the CLG description. In the main he used CLG as a design rather than an evaluation tool.

Once again the fact that O4 did not have a user evaluation laboratory may have been significant in the selection of CLG as a means of helping to ensure that the design of the UI was more user-oriented, irrespective of user evaluations. E did not believe that usability metrics were associated with the technique, although this is not strictly true. He stated that "... there aren't any, but there's no reason why people shouldn't be able to add to the notation ... for example to estimate how long certain operations are going to take." However, Moran does address the use of usability and performance metrics with CLG (Moran 1981; p 39 and pp 46 - 47) With respect to simplicity, the number of rules in CLG is suggested as a metric; "a system requiring fewer rules to describe it will be easier (faster) to learn and will result in fewer errors during use." For performance predictions such as time, Moran states "... given a table of times for the primitive actions [presumably derived as for GOMS (Card et al 1983)] the interaction methods should predict the times to do the tasks." However, it seems that these facets of CLG have never been illustrated in practice.

**Support for the Activities of the HCI Specialist, and Failure to Provide Support.**

E was able to describe some of the ways in which he felt that CLG supported his activities "I feel there is some value in distinguishing between levels (which some people don't) because I have used CLG to specify an existing system and you do get different things at the semantic and syntactic level. If you do a mapping between these levels, and use any mismappings you get as an error predictor, you do get some reliable predictions from the specification." He also stated that it provided a "bedrock" upon which to base his descriptions of the precise requirements he had for the UI to the application expert programmer.

On the other hand, E noted that CLG, like TAKD does not guide the acquisition of the task related information which an analyst would require to drive the task-level specifications in the first place. In addition t appeared to him to be weak on system
output "... it doesn't give you any support for direct manipulation or graphics type interfaces ... it says nothing relating to the presentational aspects, except to say you can throw this up in this or that display area." Of course the research behind CLG was carried out before 1981 when graphics UIs were a rarity, and the only widely available mode of interaction was command driven, so this criticism is perhaps a reflection of the age of the technique.

Many of the problems of CLG would appear to result from the lack of any automated support tool which would help the analyst to manage the complexity of CLG in non-trivial UI specifications. When asked if there was anything he would have done differently in the project with the benefit of hindsight E stated that he would not have used CLG "unless there was a tool available to help you do it. Maybe I would also have said let's not go any further until we have the UI designed, and we have a specification, in whatever form, even if it's just structured English, which says what the behaviour is going to be."

6.5.6 Summary

In the preceding descriptions each of the scientific HCI and user-oriented techniques described by the interviewees in the study has been viewed with respect to its general usage, the constraints and problems associated with its use, how it encourages or enhances exploitation of user-oriented information sources, and its ability to support the activities of HCI specialists. Notable is the diversity in the focus, scope and character of the techniques used. It would appear unlikely that a single technique would be capable of addressing all of the aspects of design covered by those revealed in the interviews.

It is clear that HCI DETs of the type discussed in Chapter 2 are a rarity even in the experience of HCI specialists, with only one organisation, O4, employing HCI specialists who used them. It may also be significant that O4 was the only organisation without experimental evaluation facilities, which could have meant that it was more important to ensure user-oriented design, since early or even late user evaluations were not possible. Another important finding is that those HCI DETs which were applied in O4; TAKD and CLG had many problems associated with them which need to be addressed by future similar approaches.
6.6 General Discussion

6.6.1 Working Regimes, Activities and Roles

The working regime of an organisation was seen to be influential in determining the activities and roles of its HCI specialists. In organisations which were hierarchical in their organisation and communication, specialists tended not to communicate directly with specialists in different disciplines. This was seen to encourage greater use of written communications which included informal evaluative commentaries, as used by C and D in O3, and guidelines for general distribution as in O2. On the other hand, in less hierarchical organisations such as O1, direct communication and collaboration between diverse specialisations and integrated working regimes were observed, as well as written communications.

The structuredness of the working regime may have been influenced by the degree to which design and development activities were repetitive over different projects in O1. However, this was not the case for E and F in O4 where the structuredness of their working regime may have been related to their frequent usage of scientific methodologies. In O1 where routine activities dominated the specialist A’s work, his roles within the design process were seen to be well defined. In O3, where work was less structured and scientific techniques other than experimental evaluations were not used, C and D tended to work in a much less formal manner, and often based evaluations on their own intuition and experience, rather than on scientific analysis.

The variability of the working regime may have been related to the variability of the projects undertaken by the organisation, for example in O4 both research and commercial projects were undertaken for a wide variety of applications. Variability might also be caused by a shortage of HCI specialists within the company. In O2, specialist B was required to assist on various research projects, write guidelines for UI design, consult on commercial projects, to set up a usability laboratory, and to run experimental evaluations.

The working regimes illustrate an apparent fact that organisations may impose considerable constraints upon what is possible for HCI specialists trying to improve the usability of products of design projects. Specialists C and D in O3, and E and F in
04 were observed to be under the same working regime with their colleagues in the same organisation, defeating the possible hypothesis that personal preference might be the cause for the variability in working regime. Furthermore, many of the comments made in the interviews indicated that various improvements in UI designs recommended by some of the specialists were not brought about because of the way design projects were run by their company. For example, where late evaluations were enforced as in 02 where specialist B complained about being restricted to late evaluations, experimental evaluations may have had little impact on the released version of a product. It may be partly the responsibility of an organisation, and its working practices, that user-oriented design and evaluation is under represented in final products.

The activities involved in the work of commercial HCI specialists (see table 6.3) can be characterised as belonging to at least one of the following four types:

- Information Collection
- Invention
- Analysis
- Communication

Whatever tools or techniques the HCI specialist uses in commercial practice must have some value in supporting one or more of these types of activity if it is to be of any practical use. The more types of activity a technique supports, whilst the effort of applying it remains constant, the greater its value will be.

Five HCI-oriented roles were identified for the specialists, each involving a variety of different kinds of activity (see table 6.4).

- Self Educator
- Designer
- Researcher
- Consultant
- Agent of Technology Transfer

The roles of HCI specialists and the various activities they involved were diverse, especially for B, E and F in 02 and 04 (see table 6.5). In 02 diversity of roles appeared to be a result of the large size of the organisation which B had to work for. There were not enough HCI specialists in 02 to allow specialisation in a particular role. This may explain the importance the company attached to research into technology transfer which was immediately implemented adding more weight to the
role of technology transfer of specialist B. In 04 the design projects undertaken were very diverse, the company produced software for a variety of applications which would run on other companies’ hardware.

Both 02 and 04 allowed their specialists to conduct government supported research, which may also add to the variety of activities undertaken and a strong requirement for self education of the specialists. In 01 and 03 there was greater emphasis on the role of commercial consultancy within the organisation. It is interesting to note that in 02 and 04 where research seemed to be encouraged, HCI specialists’ design activities were more directed towards design of HCI methods.

The different roles of the specialists, and the diversity of working environments and applications upon which they work suggest that they have a wide variety of requirements for support for their different activities. For example, in the role of designer, invention, analysis and communication seem to be important types of activities (see table 6.4) as was the case in 03 where C and D invented their system concepts, carried out informal analysis (by storyboarding scenarios) and then communicated the resulting specifications to the academic software research and development team. Tools and techniques of greatest value would support as many of these activities as possible. However, given the difficulty of extending scope of techniques without losing depth (Barnard 1985) it might be more realistic to recommend that developers of future techniques concentrate on providing a complete support tool for one role (i.e. a coherent subset of activities) only. A tool, supporting HCI specialists in the role of designer, which represented the requirements (or problem space for solutions) of the system in such a way that invention, and analysis of solutions was easier, and in such a way that communicating with others was easier, would be more valuable than one which simply represented the requirements for the system. The nature of a valuable tool for a consultant might have to be different, perhaps putting more emphasis on support for the information collection which would drive a requirements specification.

Of course determining the nature of the right tool for the job depends very heavily on the precise nature of the activities (see summary of activities and roles) which tended to vary even within the roles identified, for example the role of technology transfer agent might involve preparing a simple methodology which could be taught to non-HCI specialists (Design/evaluation technique development), writing
guidelines of some sort (Writing reports/documentation), or personally teaching the technology (Running workshops, giving seminars or presenting papers). No single technique or tool is likely to support all of these activities. However, without a clear picture of the variability of HCI specialists activities, their requirements for support, and the diversity of the nature of the applications that the transferred technology would have to adapt to, it may be difficult to design tools or techniques which are neither too generalised nor too specific and unadaptable. In the following section the findings relating to the important features of the techniques applied by HCI specialists in commercial practice, are discussed with respect to what made them more or less applicable. In this way it should be easier to identify how applicable techniques do, and might better, support the activities of HCI specialists, and possibly non-specialists also.

6.6.2 Important Features of Applied HCI and User-Oriented Design and Evaluation Techniques

In this section the descriptions of the five scientific techniques used by HCI specialists in this study are summarised in terms of the requirements they have for successful application; their ability to cope with commercial design constraints and problems; how they encourage or enhance exploitation of user-oriented information sources; and finally the manner in which they support the activities, and hence the roles of HCI specialists. A further brief discussion is devoted to consideration of the reliance of the techniques observed upon the skills and experience of HCI specialists in particular, as opposed to non-specialists who might benefit from such techniques.

Requirements for Successful Application of the Techniques

All of the techniques described by the interviewees required some investment in terms of time and/or expense. For experimental user evaluations the investment was particularly heavy, since in all cases valuable space and equipment was required as an initial investment. The technology transfer technique developed in O2 required research, and then repeated training sessions. The three description and analysis techniques (Statecharts, TAKD and CLG) were, as applied by the specialists considerably cheaper, but still required considerable self-education and time to carry out (CLG probably being the most complex, and requiring skills in HCI). It is
not clear what the precise costs were in terms of time and money spent, as against the benefits of the investment; this is probably difficult to work out; Mantel & Theory (1988) have proposed a methodology for cost benefit analysis but have not demonstrated its ease of use in practice.

However, what seemed surprising was that in several instances, but nowhere more clearly than in the use of experimental evaluations, organisations seemed to under invest in support tools according to interviewees. In experimental evaluations, the problem was particularly clear because HCI specialists in O2 and O3 were testing products which could not be significantly modified before release, and many of their recommendations were ignored or deferred for later products. Had they been involved earlier in the design and been able to use a UI simulation tool as in O1, to prototype and evaluate the UI, the same effort which went into modifying later releases of the product could have been applied earlier on the basis of experimental evidence from user evaluation of a simulation. The under investment was also evident in the purchase of support tools for Statecharts specifications which specialists A and F were spending a very long time preparing by hand. The problem seemed to be one of justification of expenditure when nobody knew what the costs and savings might be for investment in new technology.

Another requirement of the techniques identified in the interviews was that they integrate well with the host organisation's working regime (which tended not to be easy to alter). The technology transfer technique had to be simple enough to be passed onto a design team syndicate in a two and a half day workshop, because O2 did not allow more time to be spent with the recipients of the technology, and the recipients were not followed up or supervised afterwards. TAKD and CLG were weak in this respect because they expected more investment in terms of information collection than was possible, and they did not work as communicable specifications which could be understood and used by other members of design teams who would have to implement their recommendations. Systems programmers never had the time or incentive to learn how to use these notations, nor was there any guarantee that the specifications would have been sufficiently clear or appropriate enough to support design activities. As stated previously with TAKD the notation loses much contextual information which may be crucial in directing appropriate functionality and UI dialogues. CLG specifications, generated by HCI specialists, are not sensitive to software limitations which may only be known to programmers who cannot
understand these specifications.

If an organisation's working regime isolates various specialist groups from each other, as seemed to be the case for most of the interviewees, then the most important requirement for an integrable technique may be its power to support communication between specialists with very different backgrounds.

Another issue strongly related to integratability is that of the timing of the technique in design projects where many activities take place in parallel. A technique, such as CLG, or experimental user evaluation may depend on other design activities such as information collection, task analysis and user requirements analysis for CLG, and generation of a working prototype for user evaluation. It is important that those depended on activities and the activity of applying the technique itself typically take place before the results of the technique's application begin to become redundant, due to the increasing fixedness of the ongoing design. It is probably unrealistic to assume that working regimes can be adopted to suit the requirements of every new technique that comes along. It is likely that the most applicable techniques in the foreseeable future will have to suit existing working regimes and design approaches, no matter how imperfect these regimes seem.

With respect to dependence on other implicit design and evaluation activities, the technology transfer technique depended on a special in-house analysis technique called a market segmentation, Statecharts depended on realistic task, and functional specifications (otherwise their precisely specified content would be verifiable but invalid as far as the appropriateness of the design went), TAKD depended on a great deal of information collection, and CLG depended upon task analysis, and application of knowledge of how to go about UI design (which was originally planned as a set of guiding rules to go with the technique; Moran 1978). None of these depended-upon activities were specified by the techniques themselves and numerous problems or complaints arose because of this, the worst being related to the technology transfer technique where much time was wasted rewriting a design syndicate's market segmentation because they had done it wrong. It would seem that applicable techniques ought to be self sufficient in that they should not depend upon unspecified, unguided activities, or at least they should specify other suitable techniques which must precede their application.
Most activities of HCI specialists, including the use of scientific techniques, involved resorting to their skills, experience and inventiveness on many occasions. The techniques applied were obviously reliant upon these abilities, since they often had to be adapted or viewed in a particular way to deal with specific requirements of various projects. Experimental user evaluation must be the most idiosyncratic in terms of its variability as a technique. A notable invention was the use of two subjects who, in talking to one another, revealed more about their problems than their simple errors would have.

TAKD was notable for its need of extensions, which were used to enhance contextual and sequential information as well as to determine allocation of task functionality to the intended system. Statecharts required its users to bear in mind consistency and simplicity in their specifications, for the sake of users. Without such considerations it would have been easy for them to specify unusable systems using the notation.

*Coping with Commercial Design Constraints and Problems*

Design of the UI is necessarily a process involving uncertainty about the effects of decisions in its early stages, and commitment to decisions (however inadequate) and hence inflexibility, in later stages. This poses a problem for all techniques which seek to provide predictions or evaluations of the developing system, unless the aspects which they address can be simulated first of all. Unfortunately, the facilities to simulate Uls adequately were only readily available in O1 in this study. It may not be long before UI simulation tools are a widespread commodity in HCI specialists teams, however until they are, techniques which address prediction and evaluation of Uls must be able to deal with this problem, either by tolerating great uncertainty about the actual nature of the system during early design, or by restricting claims of value in terms of their potential impact upon design. Once simulations of the whole UI can be produced before code is written, it should be much easier to see the impact of particular decisions on users and to get things right without resorting to guesswork.

TAKD and CLG require detailed information collection about users and tasks which proved to be difficult for the specialists who used them. However, even given limited access to user information, it was possible for E and F to apply them. The representativeness of their task specifications based upon limited interviews or
intuitions about the nature of tasks may be in some doubt, however it did not appear to compromise the completeness of the descriptions they produced, in terms of the range of tasks included in their specifications. Whatever technique had been used the problem of limited access to user-oriented information would have been the same, and there is probably no simple way of getting round it.

On the other hand, with TAKD, the greater the number of users providing information upon which to base task specifications, the more complex the specification became. TAKD enabled the analyst to represent alternative methods in the KRG sentences, without supporting a means for paring this number down to the most probable, or psychologically plausible representations of methods. As an acceptance grammar TAKD can express all possible task methods, but only at the expense of greater difficulty in interpretation of the notation due to its complexity. To use it as a competence grammar would have meant losing much of the richness of the variability possible for task execution. Furthermore, it was not always easy to determine which method was the most appropriate for the task, so TAKD as a competence grammar could simply be an idiosyncratic collection of methods based upon the analyst’s selection from what was observed.

A related problem was the representativeness of prospective system users for all design and evaluation techniques. D, E and F all mentioned that this was an area of difficulty since access to the most appropriate groups of individuals may be impossible for a variety of reasons.

The complexity of the system application (the most important design constraint identified in the features analysis) may exponentially increase the complexity of any specification which represents what tasks it will support or the intended functionality. This suggests that support tools may be an important requirement for future HCI techniques if they are ever to become acceptable to the majority of HCI specialists. In addition, for non-specialists, operating under the fairly common constraint of lack of experience with HCI (from the features analysis), additional guidance would be necessary to substitute for the skills of the specialist. If it is unsupported then the technique must be easy to apply. Experimental evaluations seemed to be easy for HCI specialists to apply, but sometimes difficult to interpret, as specialist D pointed out. The technology transfer technique was made very simple for the recipients to follow, but it turned out to be difficult to adapt to specific projects.
with unforseen requirements. Both A and F thought that Statecharts was easy to understand and acceptable to apply by hand on relatively simple UIs, and E and F thought that TAKD was acceptable without support tools. However, E complained that CLG was extremely difficult even for himself as an HCI specialist.

Variability of design projects means that applicable HCI techniques must be adaptable to suit changing demands. As Barnard (1986) points out, there is a depth breadth trade-off in the power of any modelling technique which includes some of the user-oriented techniques described by the interviewees in this study. Experimental evaluation, being essentially unstructured, apart from the specific design used by each experimenter, copes well with a wide range of UIs and was not described as being difficult to generalise. This may be because real people act as models of the prospective users, but they are far more adaptable than cognitive competence and performance models which have limited scope and/or depth. Statecharts, being very limited in scope in that it only claims to represent system states and transitions without judging them, also seems to generalise well. TAKD was also free from criticism with respect to generalisability, since it was only used to structure task descriptions. However, the simple UI design approach covered by the technology transfer technique in O2 was criticised for being too general for use on certain projects with applications for which many of the recommended activities were simply not relevant, and CLG was criticised for being unable to address issues relevant to graphical UIs.

Encouragement or Enhancement of Information Exploitation

All of the applied scientific techniques described by the interviewees encouraged, or enhanced the exploitation of available user-oriented information sources. One of the main determinants of whether a technique was selected seemed to be its ability to capture information which was, in the HCI specialist's view, relevant to the particular activity being undertaken with respect to improving the design approach or the UI with respect to usability. This however did not necessarily seem to be related to the views of the rest of the members of design teams. Interviewees repeatedly remarked that experimental results, or the content of specifications, were ignored by management or software specialists.

What the HCI specialists seemed to view as important information varied
extensively. Experimental evaluations were used to provide information about the likelihood of user errors, and time taken to complete tasks with the system, however the explanation of difficulties was not always as easy to provide. The technology transfer exercise was intended to pass on a methodology for encouraging and structuring exploitation of user-oriented information for requirements specification by non-HCI specialists. Statecharts were used to specify the states and dynamic behaviour of reactive systems. TAKD was used for the exploitation of information about how prospective users went about carrying out tasks which the system was intended to support, and emphasised commonalities between similar task actions and objects. CLG used task descriptions to enhance a user's conceptual model which should be supported by the system. This model was maintained throughout the top-down specification of the UI dialogue structure.

Apart from the nature of the represented information itself, a number of other potential information related features of techniques seemed to drive their selection.

Accuracy was a feature of experimental evaluations, in that, as far as those who carried out such evaluations were concerned, real people provided more accurate information about the usability of the system UI than was possible with other predictive measures. Statecharts was also selected for its accuracy since it enhanced the precision of information in specifications passed onto programmers.

Economy of the representations of information supplied by notation based techniques (i.e. Statecharts, TAKD and CLG, used by specialists A, E and F) seemed to be important. Statecharts were praised for their ability to capture quite complex information in a compact form. TAKD condensed the wide variety of task related information gathered by HCI specialists into more manageable, concise descriptions. However, CLG was seen as being too complex and possibly redundant by the specialist who applied it. CLG requires that the whole system is specified at each level of analysis, which means that the analyst applying it, or the recipient of a CLG specification must produce or assimilate four times as much information as would be necessary with a single level specification.

The expressiveness and clarity of notation based techniques was a source for comment amongst the interviewees who used them. The statecharts notation was thought to be very easy to interpret because it is visually clear, capitalising on
human perceptual and information processing strengths, and providing all the necessary information relevant to its usage, with little requirement for learning the notation. However, TAKD, although easy to learn, and relatively clear, seemed to lose information available from its sources (restricted to interviews for specialists E and F) which was relevant to its application, i.e. that of task context and task sequences. This meant that ability to interpret the notation was dependent upon familiarity with the application domain and its human tasks. The software developers who were recipients of the TAKD specifications produced by F did not have this kind of experience. CLG was probably the most criticised notation in terms of clarity, although it was highly expressive in terms of specifying precisely what was required of the dialogue for the system. The Lisp-style notation of this technique was even difficult to copy let alone interpret. A secretary in O4 had great difficulty typing out CLG descriptions from hand written versions, and other members of F's design team could not understand the specifications. The evidence from the interviewees suggests that if a notation is not both expressive and clear, then it will probably fail to act as a successful communication device between different members of a design team who may rely on the information contained within it.

Support for HCI Specialists

The scientific techniques used by the HCI specialists interviewed seemed able to support a variety of activities and roles. No single technique was seen to support all activities and all roles. For the sake of simplicity the wide variety of activities identified were classed as members of at least one of the following types: information collection, invention, analysis, and communication. These activity types had various importance with respect to the five HCI oriented roles identified for the specialists in the interviews; self educator, designer, researcher, consultant, and agent of technology transfer (see table 6.4).

For each of the techniques observed it is possible to provide a rough indication of the extent to which various activity types were seen to be supported. This assessment is not assumed to be quantitative, it merely provides a rough summary of the contribution of each technique to the activity types, and indirectly to the roles of HCI specialists. However, this summary provides no indication of crucial aspects of techniques regarding their method and provides no more than a crude assessment of scope in terms of the type of activity supported. For this reason there is no way
of assessing what technique would be appropriate for a given project, and how it might relate to the rest of the design process.

Table 6.11
Support of Scientific Techniques Identified for Various Activities of HCI Specialists in the Study

<table>
<thead>
<tr>
<th>Technique</th>
<th>Activity Types</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Information Collection</td>
</tr>
<tr>
<td>Experimental Evaluation</td>
<td>X</td>
</tr>
<tr>
<td>Technology Transfer</td>
<td>-</td>
</tr>
<tr>
<td>Statecharts</td>
<td>-</td>
</tr>
<tr>
<td>TAKD</td>
<td>-</td>
</tr>
<tr>
<td>CLG</td>
<td>-</td>
</tr>
</tbody>
</table>

X = Supports Effectively  
x = Supports Inadequately  
- = Does Not Support

If the contents of table 6.11 are compared with those of table 6.4, it should be possible to assess whether the support provided by each technique is relevant, i.e. valuable, to a given role for an HCI specialist. For example TAKD has the potential to support all of the roles identified at least to some extent, because it is an analytical
tool which enables its user to condense and transform task descriptions. However, this is not to say that TAKD will be a suitable tool for specialists in all circumstances and for all applications. That would be dependent upon the scope required of the technique. Statecharts should also be capable of supporting all roles of HCI specialists, but it clearly has a different scope to that of TAKD, and will be applicable in different situations.

Experimental evaluations, technology transfer techniques, do not support invention for HCI analysts, although they support analysis which could clarify the bounds of what inventions might be appropriate. For this reason they are not appropriate techniques for designers. Statecharts, TAKD, and CLG, as abstractions of what is known about tasks and systems, all provide a novel view of available information which may to some extent support design by clarifying areas of uncertainty within the solution space. For example Statecharts enhances the visibility of the effects of actions in states given certain conditions, TAKD highlights commonalities between objects and between actions, CLG, if its assumptions are valid, captures misappings between tasks and interactions as represented by system users. However, it would be an exaggeration to suggest that these techniques actually drive design since they only emphasise parts of the solution space, they do not themselves suggest solutions.

Table 6.11 suggests that as an analytical support tool experimental evaluations are weak because they do not necessarily assist in the explanation of users difficulties with the UI. Technology transfer does not deal with support for analysis by HCI specialists, although what is transferred may provide analytical support for its recipients. All of the notation based techniques supported analysis of one sort or another as described earlier.

Technology transfer is the method that seems to provide the narrowest type of support for HCI specialists, namely communication only. As has been noted earlier in this chapter, its main value appears to be one of time-saving. By providing the recipients with the tools of the HCI specialist's trade, so to speak, the specialist is saved from having to repeatedly undertake the same activities for the same group of people. This is an idealised view, however, and the extent to which technology transfer really works remains unclear from the interview with specialist B.
As emphasised above, table 6.11 gives no indication of the method and little indication of the scope of each technique and how these might support HCI specialists in relating to and integrating with the rest of the design process. For example CLG was seen by specialist E as useful in a top-down design approach, for the analysis and description of the appropriate structure of a grammar for interactive dialogues. Were the approach used to have been bottom up, it would have been less appropriate since the notation, being paper based rather than on line, is not sensitive to constraints imposed on possible dialogues by previously written code.

The scope of each of the scientific techniques as employed by the interviewees in this study has been summarised as succinctly as possible in the findings section of this chapter. Where the analysts used the same technique, they tended to apply it to apparently dissimilar projects to address slightly different aspects of the tasks or system. For example, Statecharts, although used in similar ways by A and F, was used by A to model the states of the underlying virtual system with which the user was interacting, but specified the dialogue separately. F, on the other hand included his dialogue specifications in the statecharts themselves. TAKD was extended in similar ways by E and F, who worked closely together, however only E used task hierarchies to add sequential information about tasks. Experimental analysis was used for all types of application in O1, O2 and O3.

By adaptation of a technique or different interpretations of what it can address, each specialist produces a unique style of that technique which may represent a subset of what was recommended by its creators, plus any extensions which the specialist sees as useful. Most notable amongst external factors which might be responsible for forcing modifications was the requirement to integrate and communicate with the rest of the design process. By omitting aspects of the technique which are irrelevant or too complex to produce in the circumstances, (for example E did not use all of the features of CLG) and by adding other features (such as the contextual information supplied with the TAKD specifications produced by E and F) specialists were able to reduce valuable time spent using the techniques, omit aspects which were not likely to add value to their specifications as far as the design was concerned, and to enhance or add new aspects which improved the technique for their purposes.

One point which should be emphasised here is that it would appear that the most
popular of all the user-oriented design or evaluation approaches was experimental user-evaluations, which also has the broadest potential scope. Its flexibility may be its strongest feature. Experiments may be poor at explaining user problems, and it may be expensive to invest in the necessary office-space and experimental laboratory facilities but the probability that the technique can be used in almost any design project (possibly too late to be of real value) may mean that companies see it as a good thing to invest in.

The scoping issue emerged again when one interviewee reported using two techniques in his design and analysis of a UI, and these techniques turned out to have complementary scopes. In other words he may well have preferred to use a single approach with broad scope, rather than worry about the depth of analysis as some HCI DETs appear to do.

The Scoping Matrix used in this chapter and Chapter 2 to characterise design and evaluative techniques does not emphasise the depth of a technique, or the accuracy of its predictions. Such details are generally presented by authors making claims about the results of experimental validations of their approach's predictions (e.g. Card et al 1983, and Barnard 1986). The Matrix emphasizes scope, which may well be the main limitation of current HCI DETs of the type described in Chapter 2, and may explain the rarity of their successful application in practice, even by HCI specialists.

In the following chapter a more detailed treatment of the nature of scope of techniques and how they integrate with the process of design as a whole will be given. For the present it must suffice to say that the scope of applicable techniques as interpreted and adapted by the individuals who use them may be as varied as the individuals themselves, or even as the projects within which they are applied.

Support for Non-HCI Specialists

It is clear that HCI specialists rely upon their own skills and experience together with creativity when applying scientific techniques to support user-oriented design activities. Non-HCI specialists such as those interviewed in the design interview study do not appear to employ such techniques, and it is therefore difficult to determine how easy it might be for them to do so were the need, or motivation to arise.
HCI DETs such as those discussed in chapter 2, and the other scientific user-oriented techniques identified here do not incorporate guidance for all of the activities which they rely upon. Referring back to the earlier discussion of requirements for unspecified activities which a technique may depend upon before it can be properly applied, it should be noted that this issue is particularly important for techniques which are intended for use by non-HCI specialists. Some techniques, for example CLG, claim to be able to support the activities of systems designers (who may be HCI naive). However, without explicit guidance for all relevant activities, they run the risk of being misused, or simply not being applied at all.

6.7 Conclusions

The analysis of the information collected in the interviews enabled each of the questions raised at the beginning of this chapter to be addressed. Perhaps the most interesting conclusion that can be drawn is that HCI specialists have very different roles in design to those of systems designers who have more general responsibilities and engage in different sets activities (see Chapter 5). One of the main differences is that HCI specialists tend to use scientific user-oriented techniques whereas systems designers do not. Amongst those techniques only two were similar to the HCI DETs described in Chapter 2, the others had very different aims and methods. These two techniques were only used by one organisation and it is possible that they are only rarely used at all. So it may well be possible for HCI specialists to apply HCI DETs (with some extensions and a certain amount of difficulty), however the majority do not, even in well resourced organisations. Therefore it seems that the requirements for specialist expertise and good resources are not the only reason for the rarity with which HCI DETs are used.

This study has provided a critique of each of the scientific applied techniques described by the HCI specialist interviewees. The critiques each addressed the above questions and provide a good deal of information as to the strengths and weaknesses of these techniques, and provide some indication of the features required of applicable techniques. If HCI DETs lack these features, it is reasonable to assume that there will be at least some degree of difficulty in their application in commercial design practice.

Finally the implications of the study reported here may be summarised in terms of
some brief answers to the original questions which motivated it.

**Question**
What precisely are the HCI specialist's roles in the commercial design process?

**Answer**
HCI specialists, or anyone assigned the role of user-oriented designer and evaluator, may often have limited responsibility for the design of a system. Of particular importance is the role of *consultant* where activities such as *information collection*, *analysis*, and *communication* are prevalent. However, the areas of concern are restricted, and generally the HCI specialist does not have to worry about programming considerations, hardware, portability and so on. This contrasts with the two earlier design studies where systems designers were responsible for the complete system, especially when working as consultants to design a whole application.

**Question**
What design and evaluative techniques do they use, what is the scope of each of these techniques, and how are they used?

**Answer**
Two out of the five approaches described in detail by the interviewees in this study were HCI DETs of the type reviewed in Chapter 2. The other three techniques were clearly user-oriented, or able to support user-oriented design, but did not embody explicit models of users' knowledge representations or processing properties. The most popular technique (experimental user-evaluations) is the broadest in potential scope (along with technology transfer), and it seems to be scope which is one of the main failings of the model oriented HCI DETs (Barnard 1986, Green et al 1987). In the case of one of the two model oriented HCI DETs observed, it was used in tandem with another modelling technique which complemented its scope.

The techniques which were used often seemed to be adapted to fit the requirements of the project in question. Experimental evaluations are easily adapted to suit requirements and the content of the technology transfer technique was intended to be adaptable, although it turned out to be inappropriate for some projects. TAKD had to be extended to cope with issues arising in one of the projects on which it was used. It seems that analysts use techniques by following the general procedures
recommended where possible, but try to extend or modify them using their expertise, where their shortcomings become apparent.

The Statecharts notation was notably popular for its clarity which made it easy to read and communicate to others. Other notations used in HCI DETs may be difficult to interpret in a way that Statecharts are not. CLG in particular was criticised for its visual complexity. An easier to read notation produced by an HCI specialist would be more likely to be accepted as a written communication device (as Statecharts was in O1) by other members in a design team. Furthermore it might be possible to get feedback from potential users if design specifications can be understood by them.

Given existing knowledge about the psychology of visual processing, it seems strange that HCI notations, such as CLG with its confusing brackets, do not avoid complex notations and exploit human visual strengths. Even if most of the information in the techniques can be conveyed only in text form, various researchers in the field of visual information processing suggest that textual material can be enhanced in terms of readability by using graphical or pictorial information which helps to organise the text for the reader and assists in top-down processing (e.g. Waller 1980, Wright 1980). If an HCI DET is to be used as useful communication tool it needs to be readable by people other than the person who carries out the analysis.

Question
How do HCI specialists using user-oriented techniques avoid or cope with user-oriented design constraints and problems?

Answer
Again analysts had to resort to their own ingenuity in dealing with constraints or problems. Where limited user or UI information was available they made the best of what they had, and applied their techniques to this. Experimental evaluations on unrepresentative users, for example, would be carried out if representative users were hard to obtain. If to-be-developed system specifications were not available, then the analyst might investigate an existing release (e.g. the existing email system used by specialist E).
Question
How do user-oriented techniques exploit information sources?

Answer
The techniques employed by the HCI specialists in this study clearly provide a framework for making the best use of available user-oriented information sources. By encouraging analysis of prospective user behaviour, specifications, simulations or prototypes of systems, representations of target tasks, and so on, they help to ensure that information about users, and usability does get into the design process, and in successful cases, is preserved through to the final product.

Some information may be lost due to weaknesses in techniques, or constraints on time and resources which prevent more detailed analyses, however it seems that, where problems which obstruct the activities HCI specialists do not emerge, useful analyses may be carried out which can provide valuable input into design, even if it is only exploited in a later release of a product.

Question
How do the techniques support the activities of the HCI specialist, and in what ways do they fail to do so?

Answer
There are a number of ways in which user-oriented techniques can support the activities of HCI specialists. They can provide concrete, and convincing evidence of usability problems, they help to organise exploitation of information, and they may provide a theoretical basis for recommending, or explaining properties or the system which are related to usability.

On the other hand there are many weaknesses related to the limited scope and generality of some techniques. The lack of support tools which could speed up their application, or enable it to take place earlier in the design process. The failure of techniques to provide complete support for coherent roles of specialists particularly in terms of integration with other design activities, and communication between stakeholders and contributors to the design.

In the final two chapters of this thesis, further discussion of the strengths and
limitations of HCI DETs with respect to UI design are considered. The HCI specialists' practice investigation complemented the findings from the two general design studies reported in chapters 4 and 5, by describing the design and evaluation techniques used by HCI specialists explicitly for the purpose of improving usability. In other words this study reflects perhaps a fairly realistic ideal which systems designers might aspire to given tools and techniques which could compensate for their lack of time, resources and HCI experience.

What this study suggests is that there are certain desirable features of such tools and techniques, such that their selection by non-HCI specialists may be determined on the basis of their scope, appropriateness or their proven success in practice. In chapter 8 the ideal HCI DET features suggested by this study are made explicit, and the findings are used as the main basis for a framework for assessing the applicability of HCI DETs. This framework addresses design practice (based upon the empirical studies reported in Chapters 4 and 5), scope of techniques and roles of analysts likely to be using such techniques.

The following chapter serves to sum up the findings of the two general design studies and relate them to the nature of existing model-oriented HCI DETs. It presents a general view of the implications of design practice for the applicability of existing techniques.
Applicability of HCI Techniques to Systems Interface Design

Chapter 7

Systems UI Design and HCI Techniques: Presenting a View of Current Design Practice as it Relates to HCI

7.1 Overview and Introduction

This chapter attempts to present an overview of current commercial and applied UI design practice as it relates to the application of HCI DETs. Current HCI DETs as described in chapters 2, 3 and 6 are presented as placing a number of requirements on design, in order that they be applied. However it is suggested that design practice does not typically comply with these requirements and may be viewed as having particular characteristics which HCI techniques need to take account of if they are to be applicable. These were summed up in Chapter 5 in a design schema which is a simple, empirically based representation of such characteristics. The HCI specialists' study is not focused upon here (although it is summarised) since, as design projects with HCI specialists involved are in a minority, it is regarded as representing relatively unusual circumstances.

Amongst other concerns regarding applicability, it is suggested that projects may be characterised in terms of the scope of the main user-oriented issues which need to be addressed in design and evaluation of the UI. Such a characterisation can be mapped onto a scoping of an HCI DET in terms of the aspects of a system it can address. Finally, questions raised by the review of various design views in Chapter 3 are addressed in the light of preceding chapters in this thesis.

The role of this chapter within the thesis as a whole is to present a basis for discussing more positive responses to the problem of inapplicable HCI DETs. The following and final chapter will seek to present a framework for assessing and possibly guiding the design of future HCI DETs, not in terms of their theoretical direction, but in terms of qualities which are required of them if they are to be applicable in typical design projects. Much of the information underlying the framework's view of desirable features of HCI techniques is drawn from the HCI specialists' interview.
study which is therefore more appropriately reviewed in Chapter 8.

7.2 Resume and Application Requirements of HCI DETs from Chapters 2 and 3

The application requirements of HCI DETs are assumed to be any aspects of these approaches which place additional demands on a design project or limit their generality (to different types of UI and application), their appropriateness (to the types of analysis required for the system), and scope (i.e. the principles of usability and evaluation factors addressed), with respect to actual design practice. The descriptions in Chapters 2 and 3 suggest a number of interesting features of HCI DETs, including their scope, and how they view the rest of the design process into which they will be integrated (if they have such a view).

A Brief resume of the main features of each of these techniques described in chapter 2 as regards their possible use in design and evaluation follows. These resumes are based more upon the claims of their creators than on any other objective evidence from examples of applications of the techniques in practice by people other than the creators themselves.

7.2.1 Block Interaction Models

Supports: Analysis of the prospective user's knowledge relating to a task or problem. BIMs should be capable of assisting in the selection of appropriate detailed modelling techniques which might capture the most important types of knowledge relevant to the design in question. They may also support the designer or analyst in specifying how the different types of knowledge might interact, and how this might influence user performance and indicate ideal properties which might avoid user problems related to this interaction.

BIMs are potentially capable of indicating whether the knowledge required to interact with the system (ideal knowledge sources) is compatible with users existing knowledge. They are not intended to specify the details of the knowledge sources involved, however within the BIMs framework it should be possible to find detailed techniques which would enable he analyst to do so. For this reason a BIM cannot itself capture simplicity or any formal property of a device or UI. It is possi-
bly capable of indicating compatibility and UCTDs, although without the use of a more detailed analysis, these principles could only be intuitively dealt with. The model will not predict quantitative or qualitative aspects of user performance with a system. It is up to the particular analyst to select models or specification techniques which will permit this.

Application Requirements:

From the Analyst: BIMs require psychological and HCI expertise and the ability to select and apply appropriate detailed modelling or specification techniques. The BIMs elucidation of the nature of interactions between various knowledge sources, and their effects on human performance, is not presented in sufficient depth to permit a psychology-naive systems designer to apply the technique. Presumably communication or programming skills are also required in order to ensure that the implications of a BIM specification are reflected in the design of the system and its UI.

From the Design Project and Its Environment: BIMs require that the design environment allows the applier to gather relevant information about the prospective system users, the application, the UI, and the target tasks, and all of the relevant knowledge sources which the user has and should have which may affect use of the system. It is sufficiently flexible to be generalizable to a wide variety of design projects and activities, for example it could represent real-time applications as easily as data processing ones, and it could provide a framework for conceptual specification, system generation, or for evaluation. The flexibility of the BIMs framework comes from the fact that the precise representations for each of the knowledge sources are unspecified, other than in the sense that they should support mappings to other related or potentially interacting sources within the BIMs framework.

7.2.2 Reisner's Formal Interactive Grammar (FG)

Supports: Analysis and evaluation of a competence (idealised) model of the rules a user needs to represent in order to execute target tasks using an existing or precisely specified system. The notation is sufficiently powerful to permit formal properties of the dialogue to be determined. However, the more account the analyst takes of psychological validity of the rules to be represented, by adding semantic restrictions to what were originally purely syntactic rules, the less formal the specification must
The only property which the FG captures without extensions is that of simplicity of an idealised user's representation of the UI dialogue. This representation is entirely context free (system states are not captured), so that rules which might dictate when actions are valid or likely to fail are not included. Since FG is only associated with an implicit model of the psychological properties of the user, it only addresses competence properties of user knowledge. It cannot predict performance because no processing constraints of the human cognitive components which handle the knowledge represented are made explicit. The essential function of FG must therefore be to alert the analyst to complexity of a device. However, the complexity will only be clear by comparison with other devices specified by the technique. Reisner uses indices of length of rule sentences and the number of rules as metrics of complexity.

Application Requirements:

From the Analyst: FG's reliance on an implicit psychological model of the user places a strong requirement on the analyst for psychological expertise. The BNF notation used to express the rules of the grammar is straightforward to learn but of itself makes no statement about device complexity. It is the hierarchy of rules described by Reisner, which generate the terminal strings, which are the main device for capturing complexity. Simple observation of sentence string length and counting of rules in the grammar generates metrics for system complexity which can be compared with other FG system specifications. The analyst may wish to add semantic restrictions to increase the psychological validity of the grammars, however this requires more psychological skill.

From the Design Project and Its Environment: The FG is a reactive tool, in that it requires that the system to be described already exists or is specified in sufficient detail that the BNF notation can be applied to it. Since the role of FG is essentially one of evaluation by contrast, a previous or similar system specified in FG should be available for comparison. If there is to be any benefit from such an evaluation, the design project must permit modifications to the design to be made on the basis of any recommendations on the basis of the evaluation. In the case of FG analyses this must be possible after detailed system specifications have been made, or imple-
mentations carried out.

Since BNF is a well known, formally precise notation, it should not be necessary for the analyst to add much in the way of explanation to an FG specification in order to communicate it to a programmer. FG is potentially machine readable and there is no reason why automated versions could not be used to help write dialogue specifications. On the other hand many qualitative and all quantitative aspects of human cognition are not addressed by the grammar and cannot be predicted or explained by it.

7.2.3 Task Action Grammar (TAG)

Supports: Analysis and evaluation of a competence (idealised) model as does Reisner's FG. TAG is, however more psychologically rigorous in its account of user knowledge representation, in that it accounts for semantics of interactions and has a more refined view of rule structures which might be plausible in a user's representation of dialogue with a system. On the other hand TAG is less formally concise than FG because it appeals to unspecified user attributes which strongly influence the structure of the concepts and rule schemata modelled on the user, for example real world knowledge influences the user's understanding and representation of relationships between different concepts. Because of the less formal nature of TAG, it is more important that the analyst using TAG has a well developed idea of how knowledge might be represented by users, as this has greater influence on TAG than it does on the FG.

TAG supports the same type of analysis as FG in that it assumes an implicit psychological model of the system user which influences evaluation of a context free grammar which captures simplicity of the UI dialogue. In addition TAG captures similarities (family resemblances) between groups of tasks) within single rule-schemata, and also uses devices to capture compatibility with users' existing, or real world, knowledge. A metric of complexity of a dialogue is assumed to be the number of simple-task rule schemas, discounting those already embodied in real world knowledge. Compatibility can be evaluated by counting the number or proportion of concepts, lexemes and rules which are already represented in the user's existing knowledge.
Neither TAG nor FG address the performance aspects of cognition nor the process of learning and skill acquisition, since they are purely competence models. Furthermore they do not assist the analyst in evaluating the graphical qualities of system output, since they have no explicit component which deals with the perceptual aspects of human information processing.

Application Requirements:

From the Analyst: Psychological and HCI expertise is much more important for TAG than it is for the FG, since TAG resorts to many more devices based upon psychological theory (such as the feature grammar, and the representation of concepts as semantic components; Payne & Green 1986) which influence the structure of the grammar. Furthermore, it is unlikely that systems designers would understand a TAG specification without prior education in its use. It is therefore likely that the analyst using TAG would have to be capable of translating TAG specifications into a more comprehensible form for UI implementors.

From the Design Project and Its Environment: TAG imposes almost exactly the same requirements on the design project and its environment as does the FG (i.e. a detailed system specification). It also requires a detailed task analysis and user modelling to determine the content and structure of real world knowledge and task execution knowledge. The added value of its greater psychological scope is countered by the cost of the possible requirement for translation into some comprehensible form or machine readable versions of a TAG specification. As with FG, an existing but modifiable system, or UI specification or implementation is a prerequisite for obtaining the best value from a TAG analysis.

7.2.4 ACT*

Supports: Any evaluation or design method which seeks to simulate, explain, and predict human performance in information processing. It does not contain components which restrict its scope purely to HCI. It can account for complexity of competence models of dialogues and tasks, compatibility of knowledge required by a system, and the effects of incompatibility. It models performance, both with respect to human learning and actual task execution, and it deals elegantly with development of skill. It is entirely generalizable and can cope with perceptual as-
pects of processing as well as cognition.

Application Requirements:

From the Analyst: ACT* really requires a great deal of commitment and psychological expertise to apply. The complexity of a non-trivial ACT* model also requires programming skills in order to build a simulation of the human processor, since the size of an appropriate production system and simulation of its parallel organisation and behaviour would be practically impossible without computer support. There is a risk that production rules can be generated, in any ACT* simulation, on the basis of arbitrary decisions of the programmer. For this reason, the implementor should have, or be constrained by psychological theory and empirical support for the nature of the production rules that should be included in the model.

From the Design Project and Its Environment: An implementation of ACT* requires plenty of time and computer resources, together with all of the requirements necessary for TAG and FG, including task analysis, user information, and a system specification which would determine the nature of the production rules necessary for its operation. An ACT* model would provide reactive assessments of the impact of UI design upon users. Within itself it contains no components which could explicitly drive design.

7.2.5 Goals Operators Methods and Selection rules (GOMS)

Supports: Evaluation by modelling user cognition and representation of tasks to be executed with the system. GOMS is able to generate predictions of the speed and methods with which users would accomplish tasks using a system. It cannot account for the errors which real users typically experience, and these are only reflected in the model by expressing time predictions with an error margin (which is a degree of uncertainty associated with the added time a user might spend making and correcting errors). On the other hand a GOMS model can suggest areas of human difficulty with interaction in as much as cognitive components of users are compromised by the nature of the necessary steps required to execute tasks with the UI.

The GOMS family of models (the family consists of similar models with different
grains of analysis) only captures the rules required for ideal users to achieve competence for all target tasks, plus extra rules based upon empirical observation which reflect variations between methods users might apply. Variations between users can be captured by adding degrees of freedom to operator times and style variations to selection rules which can exhibit bias as individuals do towards preferred methods. Inappropriate or wrong operators methods are not included, and there are always enough operators and methods to ensure competence in the model (i.e. the model always knows how to achieve its goals).

Card et al (1983) suggest that GOMS can also be used to diagnose difficulties users might have with a given specified UI. GOMS is a performance modelling approach, and comes with a model of human cognitive architecture and processing constraints which can be applied to GOMS specifications to demonstrate where users processing capabilities might be most likely to prove inadequate. Once again GOMS is a reactive design tool, however it adds more precise performance explanations to problem diagnosis (e.g. lack of working memory would prohibit users from storing long lists of novel information).

The lack of any representation of users' non-ideal or existing real world knowledge means that GOMS is restricted to the principle of simplicity for the purposes of modelling and evaluation. It does not account for UI compatibility with existing knowledge users have, and how this might affect performance. The model human processor (MHP) model used by GOMS would have to be extended with respect to human perceptual processes to cope with differences in system output perceived by users. On the other hand GOMS has been shown to be able to distinguish between different types of UI input device and predict user performance with them.

**Application Requirements:**

*From the Analyst:* GOMS does contain an explicit model of human cognition which takes the onus of determining psychological characteristics of system users off the analyst somewhat, although understanding and extending the MHP should be much easier for a psychologist. It possibly requires psychological expertise in order to understand the importance of the human information processing model it embodies and its limitations, and implications for task execution. However, the main requirement for specialist skills in psychology and HCI seems to come from the require-
ment for the analyst to determine the most appropriate grain of analysis for representing a given type of system interaction. Some levels of the GOMS family of models may be more accurate in certain circumstances than others.

*From the Design Project and Its Environment:* Empirical evidence of user performance is required upon which to base the basic operator time predictions and selection of alternative valid methods for accomplishing unit-tasks. This implies previous trials with users on existing versions or simulations of the system to be evaluated or on comparable products. Representative subjects will be required for such trials. Task analyses and specifications are required to derive the goal structures for the model. Also a specification of the product itself is needed to determine what the appropriate operators, to accomplish target tasks, will be with the new system. Of course if the model’s predictions are to have any impact on the design, then any existing specification or implementation of the system in question must be modifiable on the basis of the results of GOMS evaluations.

In addition computer simulations of the tasks as executed by the GOMS model are necessary for modelling non-trivial systems since the model may need to be run many times to give an accurate picture of the emergent properties of different combinations of methods applied variously by the selection rules.

### 7.2.6 Cognitive Complexity Theory

*Supports:* The same kind of simulation-based, performance prediction as GOMS (see above) which it uses to represent task structures, however CCT has some additional attributes. CCT incorporates a more sophisticated, ACT* style (see above), production based human cognitive architecture and represents the system states and behaviour which gives it greater predictive scope than GOMS because, in addition to the complexity of the rules required to operate the system, it also addresses acquisition of skill, and the compatibility of users representation of tasks with the rules required to operate the system. CCT is thus a collection of three models; of the target tasks, the users cognitive architecture and processing, and the UI behaviour.

Like GOMS, CCT is empirically based and reactive rather than design driving. It can be applied to modules of the UI to produce mini UI component-evaluations. This is done by specifying the relevant aspect of the UI using a generalised transi-
tion network, and mapping the relevant user's goal structure to the equivalent device task-hierarchy.

**Application Requirements:**

*From the Analyst:* CCT makes many demands on the analyst which reflect its increased scope. It makes all of the demands described for ACT* and GOMS, but in addition the analyst has to selectively specify the device UI in terms of generalised transition networks, and this is not so straightforward as it might appear because by selecting to omit certain aspects of device behaviour, certain states which could influence predictions made by the model could be omitted.

*From the Design Project and Its Environment:* CCT requires that the design be capable of supporting building of a computer based simulation of an ACT* production system architecture, and a GOMS analysis which would take a good deal of time and effort. The device specifications would have to be based on highly detailed descriptions of the nature of the system being designed, but the system itself should still be modifiable if the analysis is to have any impact on the design. In essence CCT imposes the most stringent requirements on design projects in terms of the resources which must be available.

### 7.2.7 Interacting Cognitive Subsystems

**Supports:** Any design or evaluative methodology which requires a cognitive architecture from which to derive processing limitations of system users. ICS does not provide quantitative performance predictions because it does not assume any processing parameters (although these could possibly be added to the model). It has been shown to be capable of accurately predicting subtle effects, in terms of user errors or difficulties, of variations in the lexicon and structure of interaction languages (Barnard 1985, 1987). It also provides detailed explanations of users’ errors in terms of the cognitive processing components which it proposes.

The model is highly generalizable because it contains processing components which would enable it to make predictions about variations in system output, and the required form of input from the user. It also contains high a level processing component (the implicational subsystem) which would handle planning and goal direct-
ed behaviour. However the more detailed aspects of this component and some of the other components, such as the visual subsystem, need to be further elucidated. Consequently, in its present form ICS deals with the principles of simplicity and compatibility. Simplicity can be elucidated by looking at the amount of knowledge required by users to carry out their interactive tasks, and compatibility by identifying the knowledge they lack and how this lack affects their performance. The lack of detail for higher level organisation of interactive tasks prevents it from being generalised to address the presence or absence of user centred task dynamics (UCTDs) in a system.

ICS contains no recommended form of device representation and therefore does not address the behaviour of the application during various stages of tasks and the effects which particular, perhaps erroneous, actions may have at any time.

Application Requirements:

From the Analyst: ICS requires considerable expertise in psychology and HCI because it relies upon characterisation of the device in terms which highlight its psychologically important features (such as syntactic command structures) this is not clearly guided by the existing model and is left to its applier. The analyst really needs to have a good idea of where to look for complexity and compatibility problems. The details of the way in which each subsystem represents and transforms information are also highly esoteric for the non-psychologist, and have only been illustrated in detail for a subset of the subsystems in the whole framework.

ICS is really intended as a research tool, so it may not be fair to evaluate it as a design tool. At present its main role seems to be one of support for other more detailed and applied techniques which could appeal to this cognitive architecture for information regarding detailed processing properties of system users.

From the Design Project and Its Environment: ICS would require a design environment with the expertise and resources to develop and extend the model from its present form, perhaps to the extent where it could play a similar role to the one of ACT* in CCT. It is unlikely that ICS could be applied currently outside of the type of research environment within which it has been developed.
7.2.8 Command Language Grammar

Supports: A complete UI design specification, with a top-down perspective in which the entire UI is specified at an abstract level, ideally reflecting the user's task model, to begin with and then transformed, using mapping procedures (some of which are more explicit than others), through a number of levels down to the precise specification of the dialogue structure intended for the system. What the approach seeks to do is to preserve a coherent user's conceptual model of the tasks the system is intended to support throughout the design process. Moran (1978) intended that the CLG should be used together with a set of rules for user-oriented design, but these do not appear to have been widely publicised.

The approach produces a competence grammar for interaction with the system which may potentially be analysed in much the same way as other competence grammars for its complexity, by counting the rules at the various levels of representation required to generate the legal syntax for all commands, and looking at their length. CLG is also capable of addressing compatibility and UCTDs as long as the task level is based upon empirical analysis of users' knowledge. Such an analysis could have been encouraged by the user-oriented design rules mentioned above. Compatibility would be indicated by analysing the simplicity with which one level of the UI specification could be mapped to another. This is essentially representing the amount of extra knowledge, beyond what users need for the task application domain, which they would require to operate the system.

CLG is capable of representing some of the contextual characteristics of devices, for example the legal commands within a state can be made explicit. Since only the dialogue oriented aspects of system behaviour are explicitly demanded by the notation, and no proofs are encouraged to verify the specification, it may be easy to miss some of the state behaviour of the system.

CLG is related to GOMS and shares some of the same features, although the concept of level is somewhat different for the two. A level in GOMS is roughly equivalent to a level in the task hierarchies generated at the Task Level in CLG where each sub-task is a procedure within a higher level task. Moran states that performance metrics could be generated for interactions with the UI by associating times with actions, cognitive operations, and system responses in the specification
in much the same way as with GOMS.

Application Requirements:

*From the Analyst:* CLG lacks much of the explicit information which might determine its appropriate use, for example, Moran himself points out that CLG lacks generative mapping rules between the semantic and the syntactic levels of the description, and it is not always easy to determine what level an entity name belongs at. CLG also assumes that a task analysis has been carried out with sufficient skill that a representative model of users’ tasks can be generated. This places a requirement for psychological skills on the analyst.

The analyst using CLG has to do a good deal of work to understand all of the components of CLG, and needs to be extremely careful when writing the specification because errors seem to be easy to make (Sharratt, 1987). Furthermore, since the final descriptions are very complex, the person who has produced them will probably need to interpret them and communicate their implications to others, or use them to guide their own software generation.

*From the Design Project and Its Environment:* CLG relies upon a task analysis focusing upon the knowledge which the user will bring to the system, and a target task specification which will constrain the scope of the tasks the system will support. CLG does the job of mapping from the target task specification down to the specification of the dialogue. An implicit requirement from CLG as a design tool is that the system is completely unconstrained in structure before the process of design begins. Being top-down, CLG does not account for the possibility of configuring existing functionality, as might happen with an applications generator system, and it will not work with a bottom-up approach unless it is restricted to the role of an evaluation tool.

### 7.2.9 Common Attributes of HCI DETs

The above descriptions sum up some of the main features of the HCI DETs described in Chapter 2 in terms of aspects which relate to their applicability. In the following discussion, an attempt is made to generalise these features to present an overview of their support for design and their requirements from the analyst and the
design project and its environment.

Support:

If we are to take the claims of the authors of the above HCI DETs as realistic, the what we have are a number of partially overlapping but typically distinctive modeling techniques which are designed to represent information about users, tasks and systems in such a way that exploring potential problems in designs, and explaining their cause may be made easier. Many or perhaps all of them focus the attention of the analyst in directions which it may not otherwise have taken. Some like Reisner's action language may have their main value in forcing the analyst or designer to be explicit about the implications of their designs for users' tasks. Others like CCT and CLG more or less explicitly add information based upon empirical evidence about the nature of human processing or representation of knowledge, and bring this information to bear upon assessment of the UI.

It would appear that they all have a potential capability to support design in various ways. However whether the value of the support outweighs the required investment in time and skill is not clear for any of them. At present any analyst using any of these methods is taking something of a risk because the evidence presented in Chapters 3, 4 and 5 tends to suggest that none of these techniques have been proved to work for individuals other than their creators, and as Chapters 2 and 3 suggest they have only been tested in artificial circumstances if at all.

Application Requirements:

*From the Analyst:* All of the techniques discussed require the analyst to have at least some skills and experience in psychology and HCI. Perhaps some make greater demands than others, for example the Formal Grammar of Reisner is quite straightforward whereas CLG is highly complex and depends on the ability of the analyst to carry out a detailed task analysis, assign concepts to appropriate levels, derive assessments about complexity and compatibility, and so on.

The ability to communicate the content or import of the models generated seems to be inherently necessary, since most of them adopt unfamiliar notations which are subject to many implicit and subtle assumptions which may have important implic...
tions, for example; TAG assumes real world knowledge, denoted by a special symbol next to a rule schema, will have significant effects on the resulting complexity of a system for its users. If notations fail to be communicable, then the analyst him- or herself must be able to implement a UI on the basis of the HCI analysis.

From the Design Project and Its Environment: Time and expense to a varying degree are the most obvious considerations which will affect the applicability of HCI DETs. Some like CCT are likely to incur very great costs, due to the requirement for sophisticated simulations of human processing and UI behaviour. Others such as BIMs seem to require hardly any expense at all.

Only CLG can claim to support design as well as to represent a specification which can be more easily evaluated in terms of complexity and compatibility than, say, a network model of a system such as might be generated by a JSD approach (Jackson, 1983), the rest of the techniques are all essentially reactive. For this reason it seems sensible to assume that a major requirement for them is that a design must exist before they can support development. A major problem with reactive design tools is that in theory there is nothing to prevent designers from producing a completely inadequate first prototype or specification, which could be a waste of their time. Any tool which can support conceptual specification of a system by mapping users' requirements seamlessly onto optimal user-oriented solutions is likely to have an enormous advantage over one which simply suggests that an existing design conflicts with its users requirements; it will save at least one expensive iteration of the system life-cycle.

Finally most of the reactive techniques discussed seem to assume that, whatever the nature of the project they might be applied in, there will be an available system specification which will be precise enough to allow detailed analysis as occurs with GOMS, but which will still be modifiable on the basis of any recommendations derived from the modelling exercise. By contrast CLG assumes that nothing about the system is specified until the interaction level is complete. CLG does not deal with bottom-up design influences which could predetermine some of the system's behaviour and constraints.

In Chapter 3, several discrepancies were identified between explicit design views held by GOMS, CCT and CLG. Other techniques presented no view at all, and
TAG which claims to be directly applicable to design was one of these. As a reminder, the discrepancies are listed here.

* HCI DETs do not come with a clear explanation and justification of the design process into which they are to be integrated.

* Evidence of the utility of HCI DETs over competing UI design and evaluation approaches is lacking.

* Validations of the accuracy of HCI DETs predictions are carried out by HCI experts.

* Competence models assume ideal users which may be unrealistic in most circumstances.

* Few HCI DETs address the value and implications of prototyping in detail.

7.2.10 General Application Requirements of HCI DETs

It is appropriate here to sum up the implications from Chapters 1, 2 and 3 in terms of some explicit statements which can be made about the nature of HCI DETs with respect to their applicability in design. These statements will be referred to collectively as the Application Requirements of HCI DETs. Of course they are drawn from a small subset of techniques of this nature and therefore may not be a complete set, or even universally general. However, since each applies to most of the sample discussed here, it seems reasonable to assume that they are likely to be representative.

Access to Information About Tasks and Users

GOMS, CCT (and CLG if used as a performance model) place a very high requirement for empirical evidence of user-system behaviour from which to derive the basic level operator times, human processing and system response times which form the basis for their predictions. Most of the techniques require that detailed task analyses be carried out in order to capture the knowledge, goals and possibly the methods which users will bring to a system.
Appropriateness

Appropriateness really refers to the notions about user's properties and their implications for system use which a technique is able to address, and to the expressiveness of the notation, i.e. whether and how it captures these notions in some way. TAG, for example captures the notion of semantic-syntactic alignment which refers to the generalising properties of users representations such that similar semantics is best represented with similar syntax.

Appropriateness is the best way we have of ascertaining the relative value of one technique over another at present. If a technique can address certain properties of a system and another cannot then, for analysis of a system where these properties are most important to the success of the system, the technique which can capture them will be considered to be of greater value.

Appropriateness relies upon determining how well features of a technique such as its scope, input (e.g. information derived from task analyses or empirical experiments) and its output fit the particular characteristics and requirements of a design project. For example, if accurate predictions of user performance with a system are required, then pure competence models may be of no value to the analyst.

Designers' Experience with HCI/Psychology

Since HCI specialists are rare in commercial design practice, especially on smaller projects (see Chapter 4). The largest market for HCI techniques consists of individuals with limited or no experience of psychology and HCI. By contrast many HCI DETs seem to be heavily dependent upon implicit psychological models or require specialist activities such as task analyses, or involve esoteric assumptions about the psychological aspects of the user and the meaning of certain components of their notations (such as the real world knowledge rules in TAG and the multi level abstractions of CLG). It is clear that an important application requirement of many of these techniques is the ability of the analyst to interpret them in the light of knowledge and experience with psychology and HCI theory. Evidence from Chapter 6 suggested that extensions had to be made by HCI specialists to an HCI DET. These extensions were only possible because the analysts using it were aware of its limitations and the ways in which it could advantageously be adapted. It is
unlikely that this would be true for non-HCI specialists or non-psychologists.

Existing Modifiable Specifications

Adaptable, existing system design specifications are the main or only implicit assumption about design projects made by most of the HCI DETs discussed in this thesis. CLG is the exception because it will not work with existing specifications (if it is made to do so then the main value of the whole top-down approach is lost because the order of description is essential to ensure preservation of the user's conceptual model). For the majority of HCI DETs the existing specification; be it in the form of diagrams, English statements, or a simulation prototype, is the crucial underpinning of their model. The model of users' representations or of their interactions with the system, generated from the system specification produces the metrics, or has the qualities from which their analytic inferences are drawn. Only after the implications of the derived model are made explicit can these techniques impact upon design. All of the techniques discussed have the additional requirement that any existing system specification is adaptable with respect to their recommendations. If this is not the case then the value to be derived from their application would have to be deferred until the next release or version of the design, if one does emerge. Furthermore, even if such a system specification does exist, they do not describe how to relate the technique to whatever form the specification might be in, nor how the output from the technique might drive modifications or improvements.

Communication

Apart from the scope of an HCI DET and the expressiveness of its notation, communication is an important issue for aspiring practical design techniques. A notation may well be highly expressive of certain properties, but this expressiveness may not be clear, and may therefore be difficult to communicate using the notation alone. Communicability must depend upon shared understanding of symbols and conventions, and it must also depend upon the naturalness of those symbols and conventions. Naturalness is perhaps a dangerous term to use, but by this I mean the extent to which the symbols and conventions reflect and capitalise upon the nature of human perception. Communication of design and evaluative techniques' notations depends upon both the exploitation of what people perceive most easily and on use of commonly understood symbols and conventions (which are likely to be less
common between HCI specialists and systems designers than they are within the HCI community).

This communication requirement is still implicit for all techniques which use unfamiliar notations and associated concepts, and which, due to their sophistication, are unlikely to be applied by systems designers themselves. HCI Specialists who may not carry out the actual design may be required to use such techniques and communicate the output from any analysis to the rest of the design team. The more esoteric the representation, the more necessary it becomes to actively communicate it's import to others in a more digestible form. Reisner's BNF notation is perhaps the most generally comprehensible within the domain of computing; its communicability being based upon shared understanding rather than any inherent naturalness of BNF compared to other notations, CLG is perhaps the less so because it introduces so many notational devices and interpretations drawn from HCI and psychology, and ICS is probably the most remote because it uses both an idiosyncratic notation, and refers to many esoteric, psychological constructs which make up its architectural and processing assumptions.

**Time and Cost**

Perhaps it is unnecessary to point out here that most value adding design activities come with a penalty in terms of time and cost. The important feature of this application requirement is that it should not be so great as to defeat the scope of the technique itself. For example a technique which provides late evaluation only, may add virtually no value to the design (and perhaps only a little to later releases or products); its penalties ought therefore to be minimal. Design driving techniques, which preempt the need for undirected multiple iterations of designs in the hope that improvements will be stumbled upon, seem intuitively to be more valuable than reactive ones. We may expect therefore that expense of a design driving method is not so much of a penalty as it is for a late evaluation method, since it may be more likely to save expensive iterations.

None of the HCI DETs discussed in this thesis have ever been cost evaluated, it is therefore impossible to judge objectively the penalties and value associated with them in actual system design practice. Caution could make designers exaggerate the costs and minimalise the benefits of a potential untried HCI DET which did not
address this requirement. Future HCI DETs need to be tested and supplied with explicit case history based evidence of their value. An example of this kind of effort is that of Gould et al (1987) who demonstrated the value of their design principles in practice, on a commercial style project.

Integration

This application requirement is, in fact, important for any specialist design technique which requires that other design methods or activities take place in addition to those they advise for design or evaluation. In Chapter 3 the views of design made explicit by GOMS, CCT and CLG were discussed. These views imply preconceptions about design, for example CLG assumes a top-down approach, certain activities which must be addressed are assumed to be completed (for example determining the correct level of analysis for a set of tasks with the UI is necessary but not guided by GOMS or CCT). These techniques are the best of those reviewed in this respect because they at least have a view; the other techniques do not have anything to say about design circumstances and how these would affect their use.

As well as preconceptions many HCI DETs produce output in the form of evaluation which may be qualitative or quantitative, and which may only be comparative (e.g. Reisner’s FG). Only CLG, of the techniques reviewed in Chapter 2, produces a design specification. The output of the various techniques may not be in a form which is readily assimilated into the design process and the analyst may have to translate concepts and measurements into more readily understood forms, for example consistency may have to be redefined as reuse of low level components of code, and simplicity may be defined as detuning and diffusion of modules of code which underlie functionality such that dialogue commands are generalizable and not compounded unnecessarily.

The requirement placed upon the analyst is to find a way of fitting in with the rest of the design process. If an HCI DET is flexible in what it can support, and the value it delivers, then it should be possible to adapt it to suit the circumstances; for example Morton et al (1979) who developed BIMs make no claim as to the precise nature of the framework’s application, but do not constrain the analyst to a particular approach. On the other hand they have not been shown to work well with any approach in particular. CLG on the other hand is very highly constrained, and will not
work with existing specifications. An HCI DET must integrate well with other design methods, otherwise it disqualifies itself as a partial system DET which cannot work with other DETs to produce a complete methodology. The degree to which a technique is integratable is probably dependent upon how well it addresses its other application requirements such as its dependency on HCI expertise or the amount of time and effort it is likely to take up.

These application requirements are important indicators of how likely it is that HCI DETs in their present form can be applied to commercial design practice. We also need to look at the state of affairs in typical commercial design projects in order to determine whether it does meet these requirements, and in what ways it fails to do so.

The two design studies reported in Chapters 3 and 4 were able to supply this type of information. In the following section, the main findings of these two studies will be related to how the projects involved did or did not meet the requirements for application of HCI DETs. The two studies also combined to provide information about the constraints which limit the ways in which design can be practised, and the user-oriented information sources which are commonly exploited in commercial practice.

7.3 Review of Main Findings from the Design Studies in Chapters 4 and 5 and their Implications for HCI DETs

7.3.1 The Features Analysis of Design Practice

This study set out with a number of hypotheses based upon the discussions in chapters 2 and 3, and a number of informal discussions with systems designers at Queen Mary College where the research took place. These rather informal and general hypotheses were based upon the empirical studies reviewed in Chapter 3 and on some informal discussions with systems designers at Queen Mary College, London. They are directly related to the application requirements of the HCI DETs discussed in Chapter 2. For this reason each of the hypotheses and how it was supported will be discussed with reference to the application requirements of HCI DETs.
1. The design process is very variable, in terms of activities and organisation.

One of the notable findings of the features analysis was the enormous variability amongst design projects described by questionnaire respondents. No consistent method for any aspect of the design cycle was identified; in fact informality and idiosyncrasy was the main impression provided by the analysis. Only 9 out of 25 projects (36%) reported using formal methods (respondents' definition of the term "formal" sometimes seemed to mean structured). Team sizes ranged from 1 to about 50 individuals, and project lengths from 0.67 to 1800 person months. Only 7 teams (28%) involved an HCI specialist, and these tended to be the larger teams with longer projects.

What this means for the application of HCI techniques is that the target users of these are likely to have widely differing requirements for design tools. As stated above HCI DETs require investment of time and money to a varying extent this requirement will limit the choice of those who have not got the resources to more simple techniques. This is not a problem which is peculiar to HCI however (Lyytinen 1987). On the other hand, the wide range of activities undertaken by the respondents in their projects means that any assumptions made by HCI techniques about the nature of the design process could be invalid in many instances. This issue was elaborated by the findings from the supplementary interview study which looked more closely at the quality of design practice.

If any view of design practice is to be held at all, it must be empirically based and representative rather than ideal, if it is to be effective. It must also account for the probable constraints which may impinge upon the pursuit of user-oriented design. Such a view does not seem to come with HCI DETs which may explain the general reaction to their practical demands as being one of scepticism (see the discussions in Chapter 5).

2. Applications are diverse and text editors only make up a small minority of these.

Out of 25 applications reported in the questionnaires, only one was a word processor. Another 8 were also text or file oriented. Since these applications involve simple manipulations of information or UI objects such as windows and text which require only one representation, and are unlikely to be affected by other processes or
users, it might be reasonable to propose that techniques tested on text editors could be extended to these. In fact this group included two communications systems which could have had much in common with the hypothetical "EG" system on which CLG was demonstrated (Moran 1978 & 1981).

However there were 11 data and process oriented applications which could involve information perceived by the user changing in accordance with remote input, and possibly multiple representations (e.g. tables, graphs, flow charts and text). None of the HCI DETs described in chapter 2 consider the properties of UIs to such systems. Furthermore 2 CAD, 1 complex problem solving support system, and 2 expert system applications were identified. These are also unlike text editors in their character and are not addressed by the HCI DETs reviewed in this thesis. In total 16 of the applications described (64%) were very different from text and file oriented systems, and could have had very different analytical requirements. The appropriateness of the HCI DETs described in Chapter 2 to such applications is doubtful, since they do not address complex problem solving activities, tasks requiring prolonged attention in distracting or tiring conditions, parallel activity, perceptual processing, and many other such components of human psychology which are highly important to many of the tasks carried out by people using computers in industry today.

The UIs to these applications were typically menu based with WIMPs being common amongst these. There were also a number of forms based UIs included in the reported design projects. We could expect that menu based dialogues exhibit a number of different properties from command driven dialogues such as described for EG by CLG. For example, the memorising and ordering problems of commands and arguments does not arise to the same degree in menu dialogues; memory requirements are lessened because the user only has to rely on recognition. Again these issues are not addressed by the HCI DETs in Chapter 2.

It is necessary to present a wider view of the psychological properties of users than has been the fashion so far in HCI techniques. Indeed the scoping matrix introduced in Chapter 2 does not give sufficient detail on these issues, and at present may be inadequate for representing the real scope of HCI DETs because it fails to highlight which properties of the user are captured by the models. In failing to do this it cannot, for example, differentiate between two techniques which both model skilled task execution but one of which is able to model perceptual skills and might
be more suitable for modelling medical diagnosis from radiographs.

3. **Prospective user populations are highly variable, and the existence of ideal or even expert users cannot be relied upon.**

All of the respondents to the questionnaires were able to identify target user groups for their applications. The majority of target users (at least 18 out of 23 different target groups reported) were clearly not computer specialists, and many may have received little or no training with the system. Therefore the assumption that competence models could be representative of users is seriously questionable. At present no studies seem to exist to show that untrained system users ever learn the most efficient task methods which competence models tend to adopt. Nor have the effects of user deviance from ideal methods been examined with respect to the effect that this has on the accuracy of the predictions of competence models. As mentioned earlier Young & Maclean (1988) recognise the possible importance of individual user variation from an idealised competence model of task execution and have presented a method for determining how users might select particular optional methods in given situations. Such a method seems to be an important component for valid predictions generated by competence models.

4. **Time and project resources are frequently insufficient for a satisfactory amount of work to be done on the system design itself which affects the UI design also.**

Before they reached the part of the questionnaire which asked them about time constraints, 8 respondents (32%) spontaneously stated that they had too little time to complete their project to their satisfaction. Inadequate resources were ranked as the fourth most important constraint with 9 respondents adding extra constraints to the list they were asked to choose from which were also essentially resource limitations.

Satisfaction is obviously a subjective judgement and is no indicator of the actual success of a project which was impossible to determine from the responses to the questionnaire. However the general implication here is that designers tend to feel that they cannot do all of the things they would like to improve their design. This implication is partially based upon the strong suspicion that the priority for most systems designers, who are generalists rather than specialists in some field, will be to ensure that all of the necessary functionality is supplied, and to make sure that the
system will not crash. There will also be other considerations such as security, modifiability, maintainability and so forth. User issues, and the techniques which deal with them, have to compete with other concerns in commercial projects under pressure. The importance assigned to them can only be improved by convincing evidence of the costs of addressing them versus the costs of ignoring them. If there is no incentive to improve usability, investment in its cause may be minimal. HCI DETs must therefore be supplied with convincing evidence of their potential to save time and expense on the design, maintenance and training of users of systems.

5. Design teams frequently lack HCI or psychology expertise as would be required for instance to identify appropriate levels of analysis in a multi level HCI DET.

Only 28% of the projects in the study involved one or more HCI specialists, and the impact these individuals had may have been limited (two HCI specialists who completed the questionnaire claimed that they were ignored to a certain extent in the projects they described). In the majority of the projects (72%) no HCI specialist was involved, and although some of the respondents claimed to have some or a little experience with HCI it seems unlikely that they could have had the kind of experience in psychology which seems to be a major application requirement for many HCI DETs.

Simpler, easier to understand HCI techniques may receive a more enthusiastic reception from commercial design practice, even if sophistication and accuracy have to be traded off against this property. An alternative would be to embody HCI tools in expert systems which could be queried by designers without requiring additional HCI and psychological skills from them.

6. Abstract design specifications of any type (including systems analysis and design methods as well as HCI methods) are not commonly used.

Ten respondents (40%) claimed to have used abstract or structured design specifications (4 described using two different techniques and the rest, only one), however only 3 used task analyses, one used a user-conceptual model, and one used a knowledge engineering environment. At least 7 of the 10 different methodologies cited by respondents were clearly not user-oriented but some of these, for example state transition diagrams, could have been used as the basis for early evaluation us-
ing an HCI technique capable of recognising the impact of state behaviour on users as CCT does (Kieras & Polson 1985).

The majority (60%) of respondents used no abstract or structured design specifications and it is not clear that they could have provided the early complete UI specifications required by the reactive HCI DETs. For this reason it might be appropriate to suggest that HCI DETs either include a method for producing such a specification in the first place, or identify a recommended specification technique to which their notation, or implicit user model, can be applied. Such an effort has been described by Sutcliffe (1988) who bases an analysis of cognitive complexity on early JSD life-cycle, entity models (Jackson 1983). This type of approach could represent an attractive solution to the problem of fulfilling the application requirement of communication for HCI DETs since it makes explicit statements about the effects of the design specified using this notation upon the ultimate users of the system. Consequently the HCI specialist can exploit existing design specifications and can communicate an HCI view of the design without introducing unfamiliar notations to the systems analyst.

7. Prototyping is a very common feature of design projects.

Prototyping seemed to be the preferred method of system generation with 56% of respondents citing its use. Only 24% of the projects cited early evaluation, and since not all of these were in the prototyping group (some of the evaluations were carried out on concept tests or on mock ups in projects where more formal approaches were used) it would appear that prototypes are perhaps under exploited as a means of allowing evaluation to take place.

Many HCI DETs do not seem to capitalise on the possibility of user evaluations with prototypes. They tend to require that specifications are produced (from perhaps top-down early representations, or based upon an existing prototype) and that these will then be analysed using the notation independently of any useful input which might come from user evaluations. An obvious example of exploitation of the prototyping method would be to recommend that if some technique is used model and evaluate a UI, it should focus in particular on areas where users had the most difficulty.
8. **Designers are not rigorous in ensuring user-oriented design and tend to favour casual approaches and late evaluation, rather than user-driven design and early evaluation.**

As stated above, only 24% of the respondents in the study described early evaluations in their projects. The majority used iterative evaluations or late evaluations with 16% reporting no user evaluation at all. For 52% of the projects the approach to design and evaluation was informal, suggesting that what evaluations took place would have been unstructured and possibly unfocused. This implication was somewhat substantiated by evidence from the ranking of constraints and the descriptions of the respondents. Constraints reflecting a lack of structure and rigour were ranked as more important than those reflecting too much in the way of structure.

It may be that designers would prefer to use structured techniques, including user-oriented techniques, if they could find ones which reflected their needs. Comments relating to the impracticality of both HCI and other structured techniques were made by designers, and their objections may be focused on the form rather than the intent of these methods. Unfortunately the questionnaires did not go further towards revealing whether this possibility was indeed the case. However the overall impression seems to be that integration and communication may be major problems with HCI DETs as far as their application requirements go. Designers may think that structured design and evaluation techniques, particularly HCI techniques, are not practical tools because they are too complex and abstract, and take no account of what the real problems in design practice are (the evidence from both the features analysis and the supplementary interview study tended to confirm this view). Again this implies that systems designers may be prefer simpler and informal HCI and user-oriented design and evaluation techniques such as that recommended by Jorgensen (1984) rather than rigorous but complex ones.

9. **There are many constraints which could pressurise design teams towards certain methods and away from novel unproven methods.**

This hypothesis justified one of the most important and revealing parts of the features analysis of UI design practice (see Chapter 4). The more important constraints seemed to represent lack of information or technological and human resources. It was also the case that constraints were more likely to be imposed by
lack of structure rather than lack of freedom. The most important constraint was seen to be the complexity of the application itself. This may have been because finding any solution to requirements was difficult, let alone finding a good design solution. If this tends to be the case, the additional burden imposed on time, technological and human resources by HCI DETs could be seen by many design teams to be prohibitive and this effect could be exaggerated where designers already have little knowledge of, or confidence in HCI, as a discipline. In other words we cannot assume that HCI makes sense to designers; it will probably have to be sold or explained to them, which implies that its benefits will probably have to be proven and made explicit.

Two further features of major interest amongst the findings of the features analysis of design practice were:

The nature of user-oriented information sources used in commercial design practice, and how they are exploited.

It was clear that design teams varied greatly in the number of different user-oriented information sources they exploited. Respondents claimed to use anything between 1 and 17 different sources (however some of these could have been overlapping). Observation of users trying out the prototype design seemed to be the most common source, however user-task specifications (these might be supplied by marketers or the design team itself) seemed to be considered more important than observations of users as a source of information. Experimentation was ranked as a valuable information source but it is not clear whether scientific experiments or simple try-it-out sessions were the norm (the definition of the term "experiment" should have been clarified by the questionnaire, but unfortunately it was not).

The main implication from the responses about information sources was that HCI techniques were hardly exploited at all. HCI task analyses were the 15th most important out of 18 ranked sources and surveys and reports on human characteristics were the least important of all. On the other hand designers did seem to be using scientific references on human behaviour; this was the 8th most important information source (for example reaction times and colour perception information was sought by designers from this source).
However it was clear that the favourite information sources were drawn directly from observations, and interviews carried out by designers themselves, and specifications and documentation on the tasks the system would support. It appears that designers generally preferred to gain direct information rather than resorting to the science base when finding out about user behaviour. They used observation of use of prototypes, and of prospective users’ task activities, interviews and descriptions from prospective users and current task performers, documentation (e.g. training manuals and surveys), task descriptions, and experimentation (more or less formally). These all constituted direct information about users and their tasks with and without the new design. HCI DETs have yet to show how they can improve on this type of information. At the moment they generally seem to claim to be informative and accurate to some degree, rather than more informative and accurate than commonly used sources (this degree is usually a comparison with another HCI DET and bears little relation to practical value; for example TAG is compared with Reisner’s FG by Payne & Green (1986), and claimed to be of superior sensitivity to user-complexity).

Designers’ understanding of what represent good and bad features of UIs.

Designers were asked to describe what they considered to be good and bad features of their own UI designs. This was an indirect way of finding out what they perceived to be good and bad UI features in practice, as opposed to in the ideal. The responses suggested that, even though the majority of the respondent were not HCI sophisticates, they did consider user-oriented design issues such as simplicity, compatibility, UCTDs, consistency, and observability (although they did not always refer to these issues by the same terms). The only principle identified in Chapter 1 which designers did not mention was retrievability. The suspicion that designers have other competing considerations as well as HCI was also supported by the fact that speed, robustness, modifiability, efficiency, and conformity with in-house style (which are features which apply to the system in general, or are not specific HCI issues) were also cited as being good features of the designs.

Designers may have some intuitive awareness of what constitutes a good UI design. Their skill in this area may come from past experience with design, rather than from assimilation of findings from psychological and HCI research. We have no way of knowing whether HCI naive designers judgements of usability are inferior to those
of HCI specialists or to experimental evaluations. As well as demonstrating accuracy and value for UI design, HCI techniques also have to be able to tackle issues which designers cannot accurately judge for themselves, otherwise these techniques run the risk of becoming expensive ways of stating the obvious.

These two additional points of interest both suggest that HCI DETs need to be empirically tested and shown to have additional advantages over existing methods used in UI design practice. At present designers may consider that they will not gain anything from applying such an approach, particularly if it is time consuming and expensive to do so. Just as systems themselves have to be evaluated and have specifications associated with their performance, so HCI techniques must be tested, in practice by representative users (other than their developers) in order to provide convincing evidence of their usefulness.

7.3.2 The Supplementary Qualitative Study of Design Practice

This interview study aimed to clarify the qualities of commercial UI design practice which might obstruct the application of HCI DETs. The quality of designers' activities and how they were causally related to design problems provided additional information relevant to the application of HCI DETs in applied design. One of the notable features of the study was that the UI is typically designed in a highly integrated manner with the rest of the system. Designers may make a logical separation between designing functionality and designing the interface of a system but this is not reflected in their practice. It is therefore more precise to talk about user-oriented design than UI design. Designers seemed to deal with the functionality and the UI and any user requirements for both together.

The main goals of the study were as follows:

* To present a more integrated view of applied, commercial practice than obtained by the features analysis of design projects.

* To determine whether HCI DETs or principles are applied by non HCI specialists in commercial practice.

* To explain the design constraints and their effects which might prevent systems
designers from applying more user-oriented design techniques.

The eight design scenarios provided a more integrated view of user-oriented design practice than the features analysis. Interviewees were able to describe the sequence of events which took place during the project and to introduce interesting information about problems they experienced and how these arose.

The supplementary qualitative analysis supported many of the findings of the features analysis. Design practice tended to vary widely, with no two projects using the same method. Approaches were typically informal on the whole and no HCI DET of the type described in Chapter 2 was seen to be used. However one of the interviewees did use a set of design principles which are essentially related to the Generative User Engineering Principles (GUEPs) of Thimbleby (1984). The main additional value supplied by the interviews was that they enabled the features of design to be put into the context of a process.

A User-Oriented Design Framework was derived from the common activities described which represented a minimal design structure (see Figure 5.2). This schema provides the basis for combining the findings of both of the design studies into an empirically based design view which captures both information sources and constraints and the general sequence of activities within which they are encountered. These can be related to the application requirements of HCI DETs in order to determine whether certain requirements will not be met by typical design projects.

If an HCI DET does not require any additional activity to be undertaken, any extra information sources to be utilised, and is able to work within or ameliorate existing design constraints, it should require little investment. For each extra requirement which has to be satisfied, the technique should provide additional benefits which justify it.

This schema might be used as a basis for viewing the user-oriented aspects of a particular design project which could represent a variation on the schema (with certain activities added or elaborated, and with possibly different information sources and constraints). Such a view could contribute to a basis for selecting a particular HCI DET in preference to others which were less applicable in the circumstances. This contribution would be in the form of an indication of constraints which could
prevent certain methods being applied, information sources which are available and which could be exploited easily by some techniques and not others, and the nature and order of various design activities which might suit the application requirements of HCI DETs (e.g. existing modifiable specifications). This schema will be considered further in the following chapter.

Table 7.1
The Eight Projects Included in the Qualitative Design Practice Study

<table>
<thead>
<tr>
<th>Numbered Design Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Display Editor</td>
</tr>
<tr>
<td>2. Network Management System</td>
</tr>
<tr>
<td>3. Garment-Pattern Design-Aid</td>
</tr>
<tr>
<td>4. Educational Graphics System</td>
</tr>
<tr>
<td>5. Window Manager</td>
</tr>
<tr>
<td>6. Office &amp; Stock Control System</td>
</tr>
<tr>
<td>7. Simulation Training Device</td>
</tr>
<tr>
<td>8. Distributed Building-Management System</td>
</tr>
</tbody>
</table>

One of the main values of the interview analysis was that it provided eight scenarios of user-oriented aspects of system design practice with a somewhat qualitative perspective. These scenarios allow a closer inspection of the ability of design practice in applied environments to satisfy application requirements of a set of current HCI DETs. Such an inspection should provide further evidence concerning the likelihood of HCI DETs, such as those in Chapter 2, in their present form ever being applied in practice. On the other hand any reasons for their being inapplicable should also be addressed. In the following sub-section each of the application requirements identified in section 7.2.10 will be considered in terms of its implications with respect to the design scenarios and also bearing in mind the more quantitative evidence from the features analysis of design practice.
7.3.3 Contrasting Design Evidence with Application Requirements

The following discussion gives examples of ways in which current HCI DETs application requirements, as described in Chapter 5, are not satisfied if certain observed design conditions or constraints exist. So, although there are design situations where it may be possible to apply these techniques, they are not adequate as a collective resource for many design projects, and therefore will not be widely applicable. Table 7.1 is a reminder of the eight projects upon which the discussions relating to application requirements are based.

Access to Information About Users and Tasks

The features analysis showed that the most important user-oriented information source for systems designers seems to be specifications of the to-be-supported activity (see table 5.4, Chapter 5). In the qualitative analysis, projects 2, 4, 5, and 8 relied exclusively on indirect information, in the form of a written or verbal specification, about users tasks, the validity of which could be open to question. The psychology and experience of the users did not seem to be addressed in any detail. HCI DETs do not explicitly direct analysts as to how to exploit such information sources in their design. This could prove to be a problem because it may not be immediately obvious to systems designers what information sources and information gathering and analytic activities could satisfy the requirement for user- and task-oriented information which most HCI DETs seem to have.

Designers may think of lack of user- and task-oriented information as a design constraint which they have to accept (it was the second most important ranked constraint in the features analysis) but it is avoidable in most circumstances, particularly if the designer emphasizes the importance of such information to any other parties with a stake in the design.

Information about tasks and users was cited as crucial by several designers. Although there was universal agreement that such information was important, four projects (2, 4, 5 and 8) did not involve user and task evaluation in their development. Lack of time, organisational obstruction and inaccessibility of users all emerged as reasons for not improving user-evaluation. Since all HCI DETs, tend to require that designers find out about users and tasks, it seems that whatever the
reasons are for excluding user/task investigation, they should be resisted. As it was in this study, the amount of information of this type collected by designers was at times certainly inadequate for application of HCI DETs, as well as for any likelihood of a high standard of user interface being developed by any method. The most obvious case of user exclusion was project 5 where even late evaluations were not carried out; the resulting UI had many problems associated with it which could easily have been avoided.

**Appropriateness**

An HCI DET may prove inappropriate to the needs of a particular design project for a number of pragmatic reasons. Most of these reasons are covered by the other application requirements of HCI DETs which may fail to be met by a design project. However, it was fairly clear from the interview study that various HCI evaluative techniques would not have been appropriate to certain design projects observed, purely on the basis of the performance metrics these techniques supply. GOMS, for example, would not have been suited for evaluation of the simulation training device since users would be error prone novices, and the main criterion for user-evaluation was the simplicity and robustness of the interface, (performance times were not of any great interest for this system). However there are other reasons for the inappropriateness of various HCI DETs, which relate to the practical design constraints identified by this study.

Referring back to Chapter 2, the Usability Scoping Matrix shown in table 7.2 is intended to represent the potential scope of an HCI DET; which may be appropriate or inappropriate to the needs of a design project. In this example it happens to be CCT. The scoping matrix can also be used to represent the nature of important properties required of a UI design; these are referred to as principles, and the considerations referred to as evaluation factors affecting the manner in which principles must be analysed and the implications which they will have. For example the principle of consistency may have one set of implications for a user, another set for the application, and yet another set for the UI. The matrix in table 7.2 also represents an informal scoping of user-oriented issues in project 5 from the interview study. This project scoping will be illustrated shortly, but first it is appropriate to recollect the nature of project 5 with respect to how user-oriented issues were addressed.
In project 5, where the design team were working on a window manager for application programmers, the issue of simplicity was not really considered in any great depth. The interviewee described "pretending to be a naive user" as his way of checking that the UI was easy to use. He described writing the code in a modular fashion, in order to make it easy to modify. The system had a graphical, direct manipulation interface and the target tasks were fairly limited (concerned with manipulation of windows, accessing and selecting from menus and so on). However the designer did not really consider any usability principle in an objective fashion with respect to the possible evaluation factors.

By looking more closely at user characteristics, properties of the UI and target tasks the designer might have realised that the complicated combinations of different mouse button presses would cause confusion in system users. Different combinations of mouse presses produced different effects with certain icons, and the same mouse button could also cause different effects depending upon where on the window or background it was pressed. With hindsight and some feedback from client users, the designer admitted that the mouse buttons were somewhat overloaded.

An HCI DET which was capable of capturing user characteristics, behaviour of the UI, and the nature of the probable target tasks of system users would have been highly valuable to the designer in project 5, so long as it did not involve too much investment of time and expertise (project 5 involved a small team in which a consultant HCI specialist produced some recommendations, but the other two designers had little experience in HCI). Perhaps a highly simplified version of CCT with its useful representation of UI states (the Generalised Transition Networks) could have been most appropriate. Context free grammars, such as TAG might have been less well able to cope with the effects of cursor position at the time of actions, and the effects of preceding events upon the current state of the UI.

Table 7.2 represents the scope of CCT in terms of the principles of usability, and the evaluation factors it is capable of addressing (see Chapter 2). It also attempts to provide an informal, and perhaps hypothetical, scoping of the important aspects of project 5 with respect to usability. It is possible if necessary to weight each of the cells in the matrix in terms of its relative importance to the design.

As table 7.2 shows, simplicity and compatibility are relevant considerations for
Table 7.2
Scoping Matrix of CCT Contrasted with Analytical Requirements of Project 5 from the Designer Interview Study

<table>
<thead>
<tr>
<th>Evaluation Factors</th>
<th>Usability Principles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simplicity</td>
</tr>
<tr>
<td>Users (Us)</td>
<td>CCT 5</td>
</tr>
<tr>
<td>System Application (App)</td>
<td>5</td>
</tr>
<tr>
<td>User Interface (UI)</td>
<td>CCT 5</td>
</tr>
<tr>
<td>Target Tasks (TTs)</td>
<td>CCT 5</td>
</tr>
<tr>
<td>Acceptability of Performance (Acc)</td>
<td>CCT 5</td>
</tr>
</tbody>
</table>

NB 5 = major importance to project 5
5 = minor importance to project 5
project 5, but they are of lesser importance. This is because the designer was able to assume that the target users of the system would be experienced programmers or at least frequent users who would become more experienced as they went along. The system was being designed on a workstation for application programmers, rather than computer naive users or infrequent computer users. Users would have the time to learn complex features of the system, or would already be familiar with graphics workstations with similar features.

UCTDs are important because the efficiency of the interactive methods chosen for the system would have depended on the nature of the tasks that users were likely to carry out. For example, frequent sequences of operations would best be satisfied by direct commands or single menu items. Expert users would be likely to become frustrated by repetitious operations which slowed them down.

Observability is very important because the features of a graphics window manager tend to focus on direct feedback from manipulations such as pointing, dragging and clicking on different areas of the screen which determine the results of each action. Consistency influences the choice of combinations of clicks and the area dependent effects. Retrievability is also important because a window manager essentially controls access to different applications (e.g. shells, text editors, spreadsheet packages etc). If the user wants to get anywhere s/he has to open a window, and s/he should be able to move freely between concurrently open windows and work with different applications simultaneously.

As an evaluation factor the system application aspects were of secondary or no importance in project 5. The evaluation of usability of the system was not dictated by concerns relating to peculiarities of the applications which might run on it. The users would have generated their own applications and the application behaviour of the window manager's software itself was largely manifested within the UI which is viewed here as a much more important evaluation factor. Also user performance issues were of little importance in project 5. Perhaps an indication of potential error sites or complexity with respect to each of the principles would have been sufficient to improve the design.

In fact CCT in a simplified form could have been able to address many, but not all of the issues which might have been important in project 5. It appears in table 7.2
because, of all the approaches reviewed in Chapter 2, it has the most appropriate scope for the concerns of project 5. Its main shortcoming with respect to scope lies in its inability to deal with compatibility and observability with respect to certain evaluation factors. Compatible menu labels, icons would have to have been chosen by the designer, and the ability of the user to perceive and interpret feedback would have to have been predicted by other means. On the other hand, CCT's lack of a powerful application model is not a major problem for project 5 because the window manager had little in the way of underlying state behaviour which might alter the effects of user actions at the UI.

In view of the picture provided by table 7.2, we might suggest that CCT would have been a highly appropriate technique to use in project 5. However it is also important to take into account the pragmatic effects of design constraints on what is possible in terms of analysis for user-oriented design.

Designers' Experience with HCI

Lack of experience with UI design and lack of experience with HCI are conditions which are important to HCI DETs because of the heavy reliance on psychological terminology and user-oriented concepts. TAG and GOMS, for example, both rely on the idea that tasks can be broken down into sub-tasks, sub-sub-tasks, and so on. The level at which tasks stop being decomposed is left to the evaluator's view of what an automatic or skilled action is, in the particular circumstances. If there is to be a wide range of users with different levels of skill, doing different tasks on the system, as was the case in project 8, then deciding what a simple-task is may be extremely difficult even for a psychologist. Since seven of the designers interviewed had little or no experience of psychology and at least 5 had very little experience with HCI, it is unlikely that they would even attempt to use any of the HCI DETs mentioned in Chapters 2 or 6.

Dagwell and Weber (1983) suggested that the poor conceptualisations of users held by designers could be improved with better, readily available information rather than education. Perhaps more comprehensible descriptions of psychological theories and their relevant implications, as proposed by Hammond et al (1983) could be explicitly included as essential background information in HCI DETs to help designers who wish to use an HCI DET to make appropriate judgements about
how to apply the technique.

Existing Modifiable Specifications

In cases where the conditions or problems of complicated application domain, lack of information about users, novelty of the application and expanding task outlines exist (see Chapter 5), it may be difficult to produce a valuable specification of a complex UI to be used for the application of HCI DETs. In seven of the projects examined the UI was developed informally, expanding functionality often occurred as the project progressed and further user requirements were identified (as in projects 2 and 3) making specification difficult. Designers are unlikely to want to spend time and effort specifying systems early on, when they are not sure of what they want the interface to be like. If HCI DETs were to be applied, it would be after the prototype was almost finalised, when all the functionality planned had been added. By this time the benefits of HCI DETs could only be applied to future designs, since it would be impossible in many cases to change the current system, either because of lack of time or limitations imposed by the existing software. GOMS, for example, requires a fairly detailed system specification, so it may not be applicable early enough in design for the analysis to influence the UI design in any significant way, before unmodifiable software is written.

There is a possibility that revised design practice and use of certain types of DET could address the existing specification problem, especially where complexity is a major factor. Kieras and Polson (1985), in their explanation of UDM (CCT), which is based in part on GOMS, recommend that the device specification should be modular, to maintain flexibility and simplicity, and that it should not be committed to any particular hardware or software specification. This DET could map directly onto the extension of iterative prototyping suggested by Smith and Mosier (1984), incremental acquisition, where separate capabilities are implemented and tested in evolutionary stages, making it possible to specify parts of an interface independently and then to add them to the evolving prototype. This may make the whole process of specification and evaluation of a complex commercial UI somewhat more straightforward, and add the benefit of psychologically-based evaluation of each incorporated function as the design progresses. However, whether this will really be feasible in commercial design environments with complex system applications remains to be proved.
Design models (such as CLG) may also be inapplicable in design projects with problems such as written software constraints and expanding task outlines (see Chapter 5). If a design team is forced to adopt a particular approach, if, for example, they have to fit in with other project contributors working on software and hardware design of a system, then certain design methodologies may not be applicable. Written software constraints will mean that abstract specifications which have been remotely generated by one party, whilst software was beginning to be written by another party, will probably not be implementable. If the task outline expands, then a structured design approach like CLG may lose its main value, which is to produce a complete, top-down, structured system description. The task-, and semantic-level descriptions of the system would have little value if the functionality planned for the system was completely different by the time the interaction-level was described.

Communication

Since none of the systems designers in the interview study used an HCI DET, and since many were working alone with no need to communicate sufficiently detailed specifications to produce software from, the main purpose of communication for the designers in this study was to gain information about requirements and transmit information about possible system functionality. Specifications of the to-be-supported activities for the system and documentation on related activities were important indirect user information sources for designers in the features analysis. Poor communication was a major problem for the designers in the interview study.

If HCI DETs had been used in each of the interview study projects, and that the content of these technique's models had to be communicated to other parties within the project. Only one of the projects involved an HCI specialist who would have been likely to be familiar with the concepts and the notation of most of the HCI DETs reviewed in Chapter 2. For most of the other individuals involved in the eight projects described, the notations and the concepts they attempt to convey would have been unfamiliar. In cases where designers were working on a consultancy basis, where conventions of notations and techniques could not be set up, clients would probably not be happy with an idiosyncratic specification of the dialogue or interactive tasks.
Communication of an HCI model depends upon both shared understanding of symbols and concepts and naturalness of a notation. Shared understanding of the symbols and concepts of HCI is likely to be rare within commercial design projects where HCI specialists are a rarity. The notations used are not easy to read either. The addition of more diagrammatic graphical devices such as has been capitalised upon in the system entity modelling technique Z, pronounced "Zed" (Abrial 1980; Hayes 1987) and in Statecharts which were encountered in Chapter 6 (Harel 1987) would improve the clarity and appeal of notations. Not all diagrammatic notations are clearer than all non-diagrammatic ones. This depends on the content and quality of diagrams; Fitter & Green, 1979). Diagrams, however, may often exploit properties of human perceptual processing which text cannot.

Time and Cost

Small design teams, or lone consultants are unlikely to be able to devote the time and effort required to carry out application of one of the HCI DETs. CLG, for example, was cited by one of the college based designers as being too time consuming to be of any real use. When commercial system interfaces are complex, HCI DETs such as CLG can turn out to be prohibitively time consuming and expensive to apply, which makes the risk of inappropriate application all the more serious for the non-HCI expert designer. On the other hand Gould et al (1987) claimed that their principled approach saved time and ultimately cost in that the interface of the OMS (Olympic Messaging System) was highly robust and rarely failed. HCI DETs must also be shown to have these benefits before they are likely to be universally accepted.

Integration

HCI DETs typically involve assumptions about the design cycle (such as a top-down approach or the possibility of empirical user observations) and some of the necessary activities, which they depend upon but do not necessarily guide (such as task analyses). They also produce output which is not necessarily structured in such a way that it can be easily assimilated with the rest of the design process. The interview study involved 5 one-person design teams; the features analysis suggested that 5-person teams are probably the median size. In one person teams the requirement for HCI DET integration could have been more easily satisfied because a single
designer can only work in series, and in the interview analysis single designers tended to be working very informally. This meant that they could possibly have adapted their methods to suit HCI DETs quite easily. Ignoring other considerations for the time being, HCI DETs may be easily integrated into projects where designers are willing to accommodate application requirements in general, have satisfied any assumptions of the technique, and have the time to interpret and exploit the output of the technique and apply it to the design.

However, larger teams are likely to employ a variety of different techniques (formal or not), which could easily be applied in parallel. This means that, whilst an early HCI analysis is being carried out, it is possible that decisions about system structure are already being made on the basis of other considerations such as data types to be used or core processes essential for business requirements. The software to support this structure may also be written. Such decisions and activities may ultimately affect what is possible or desirable at the UI.

Another problem is that early application of HCI techniques must accept the uncertainty typical in the first stages of design, and late application must accept the constraining influence of earlier decisions on what is changeable. Without a clear picture of the design cycle into which they will fit, HCI DETs run the risk of producing information which cannot be utilised because it is available at the wrong time, or based upon a view of the system under development which is no longer valid, perhaps because requirements have drifted, and changes have been made to the specification or prototype.

Of course by committing themselves to a particular view of the design process, as GOMS, CCT and CLG have done, HCI DETs risk limiting their generalizability. However, the situation is made worse by commitment to an unrepresentative design view which Chapter 3 has suggested is the case for the three techniques which have attempted a view.

Integration of HCI DETs is perhaps a requirement which is dependent upon other application requirements such as scope, communication and so on, and how the technique in question addresses them with respect to the needs arising in typical design cycles.
Overall it appears that there are many conditions and related problems imposed by the design environment and activities which are probably incompatible with many current HCI DETs application requirements. It is proposed that the current HCI DETs will only be applicable in a proportion of design environments which have compatible conditions with those required by the DETs. In other words more techniques are required to cater for design environments where the existing techniques would be inapplicable. approach to interface design and evaluation in many circumstances.

The qualitative interview study of applied and commercial design practice suggested that HCI DETs, although potentially valuable to commercial design, are not applied in practice. The design environment conditions required for the successful application of current HCI DETs do not appear to be satisfied by commercial design projects. The reason for this is the existence of unavoidable conditions and constraints in commercial design which future HCI DETs should try to cater for.

7.3.4 Summary of Implications of Design Practice for HCI Techniques

In this chapter the design studies reported in chapters 4 and 5 were reviewed, and their findings contrasted with the apparent requirements of HCI DETs in order to demonstrate mismatches which may be responsible for the rarity with which such techniques are used in practice. To sum up some of the serious implications to be considered in the light of this chapter I shall answer the questions raised by the review of design views presented in Chapter 3.

Each of these questions is answered on the basis of the findings from the two applied and commercial design studies, and in the light of the preceding discussions in this chapter.

1. Are designers prepared to devote the time, effort and resources necessary to apply HCI DETs? Do they, or can they, collect the kind of data required to support HCI DETs (such as task analyses, or user characterisations)?

The features analysis indicated that systems designers tended to have positive attitudes to the aims of HCI, if not the methods, and the qualitative interview study suggested that designers do have reservations about the applicability of HCI in
practice. The answer to the first part of this question is "No". However it is clear that systems designers do put a considerable amount of effort into determining and satisfying user requirements, but are subject to many constraints and perhaps lack some of the necessary skills to carry out these tasks to the standards that HCI specialists might like to see. So the answer to the second part of the question is probably "Yes, providing appropriate circumstances prevail".

If there are unavoidable constraints such as lack of access to, and information about, prospective system users or limited resources then designers may be forced to take short cuts rather than actually collecting the necessary user-oriented information (for example talking to managers instead of directly to prospective users). The main problem seems to be one of distinguishing between avoidable constraints and real constraints. Real constraints will only hold if there is nothing that any party concerned in the design process can do about them avoidable constraints are imposed by such factors as the unwillingness of one party to provide valuable information, or to carry out some analysis activity. Such problems may occur for reasons which would not stand up to scrutiny or because of lack of appreciation of the importance overcoming such problems.

2. Are designers well enough educated regarding psychology and HCI to appreciate the importance and relevance of various techniques available? Are they able to select, adapt and apply these techniques themselves?

Some systems designers, particularly those in academia, are well aware of the existence of HCI techniques, but choose not to use them for reasons which are not to do with their lack of experience in psychology. However it would appear that in commercial environments many designers have not even heard of HCI techniques and cannot select one because they do not know about them.

It is clear that few systems designers are experts in psychology, and for this reason it is probable that, even if they were keen to use an HCI DET they might find they did not have the necessary skills to do so within a reasonable amount of time. Further research on improving the usability of HCI DETs for systems designers is necessary.

3. Are abstract specifications themselves sufficiently comprehensive to provide a good
basis for evaluation since they do not convey the look and feel of the system to designers or users?

Abstract design specifications of any type (HCI, formal, SADT) seem to be less popular as design and evaluative tools than do prototypes (see Chapter 4; the features analysis of design practice). None of the questionnaire respondents in the features analysis or the interviewees in the qualitative study reported using an abstract specification as a user-oriented evaluative tool. Given this fact and the absence of comparative studies of HCI specifications and empirical evaluations, it is hard to answer this question. Further studies need to show that abstract specifications have a value adding role, even if it involves combining them with empirical evaluations, in providing more informative and accurate critique of system designs.

4. Can levels of analysis required by an HCI analysis be identified a priori to analysis itself?

The ability of designers to determine in advance an appropriate level of analysis for an HCI design or evaluative technique has not been demonstrated either in theory or in practice. HCI DETs define, only vaguely the precise meaning of a level. Levels may refer to the grain of analysis in terms of some dimension such as time, or implied complexity. For example a high level goal is more complex to satisfy than a subgoal which subsumes it; this is the case for GOMS. Levels may also refer to the degree of abstraction from concrete behaviour, for example in CLG a task level description may say nothing about the actual behaviour of the user with the system. This distinction is somewhat blurred and levels of one type may be the same as levels of another, for example, the interaction level in CLG is similar in some respects to a keystroke-level GOMS model (Moran (1978). In addition the definition of a unit action (e.g. a terminal symbol in Reisner's FG, a simple task in TAG and an operator in GOMS) tends to vary between techniques, and within techniques.

Given the variation, and frequent uncertainty of what precisely constitutes a unit action in an analysis, together with the diversity of definitions of level of analysis, designers may find it extremely difficult to decide what constitutes an appropriate level of detail or abstraction for a given analysis in their own circumstances. Card et al (1976) found that not all levels of GOMS were equally
accurate for text editing tasks, but they did not show how one might determine in advance which level would have been most accurate for a range of diverse task types. Given, then that this is a theoretical problem for the developers of HCI DETs, how can psychology naive systems designers be expected to tackle this aspect of an analysis. Given their suspicions of HCI DETs in the first place (see Chapter 5), they are only likely to be further put off applying them by their lack of clarity about how to identify a meaningful task, action, or some other psychologically valid component of an analysis.

If techniques do not supply this important information, then designers may have to waste time trying a number of different levels of analysis with an HCI DET before finding one which is most accurate for their particular application. This would constitute another disincentive to using such a technique.

5. Are there HCI DETs suitable for application to most design projects, in terms of user characteristics, UI type, application, and target tasks? Are the metrics supplied by HCI DETs likely to be widely applicable for UI evaluation?

This question has largely been answered in the preceding discussions. It is clear that, without considering any other issue to do with pragmatics of application, HCI DETs seem to have limited scope, and all have areas which they fail to address (for example CCT does not address compatibility of a system with users' declarative knowledge, so ascertaining the best set of command names for a system would not be supported by the technique). In addition there are many application requirements for HCI DETs which design projects do not typically satisfy, such as expertise in psychology and adequate resources. Each HCI DET is likely to be applicable in only a small proportion of projects. Furthermore the generalizability of many HCI DETs from the type of application they were demonstrated on to other systems is seriously questionable.

6. Can HCI DETs in their present form be integrated into current software design practice, and if not how can they be adapted for integration?

This final question was not addressed by the design practice studies because no information on the use of HCI DETs was collected. The third study, reported in Chapter 6 of this thesis looked at the activities of HCI specialists in commercial
environments to determine whether they were using HCI DETs (HCI specialists tend to work in better resourced organisations and have some background in psychology).

7.4 Conclusions

In this chapter a number of application requirements which HCI DETs impose upon design projects were described. The two design practice studies reviewed provide information about design practice which contrasts with these application requirements. It appears, from the evidence obtained, that design practice as it occurs in commercial and applied projects, does not naturally satisfy the expectations of HCI DETs in terms of the resources, expertise, and flexibility which would be required to apply them in many cases.

7.4.1 Problems of Inapplicability

Information about users and tasks is not always easily accessed, and the guidance required to exploit this information effectively tends not to be supplied by the HCI techniques themselves. Appropriateness of HCI techniques is not guaranteed for all design projects. An example of one of the design projects from the interview study was characterised in terms of its probable analytical requirements, and it was suggested that CCT had an appropriate scope to satisfy these requirements. However the results of this scoping exercise were somewhat confounded by the fact that the practical implications of undertaking a CCT approach would have been extremely problematic for the design team with their limited resources and constraints.

Designers' experience with HCI may often be so limited that understanding a technique alone, let alone applying it, could prove difficult. HCI techniques do not seem to be presented in a simple easily digested form which could make them accessible to any systems designer. Existing modifiable specifications, from which to derive some HCI model, may not be available because of various constraints such as those imposed by the previously written software which constrains what may be modified, or by expanding task outlines which may make early detailed specifications unrepresentative.

Communication of the contents of an HCI model and its implications may be
difficult without interpretation into terms that software specialists can understand. Notations and concepts which they embody may be obscure and confusing to non-psychologists or HCI naive designers.

*Time and cost* required when taking a rigorous approach to user-oriented design may be prohibitive for small or poorly funded design teams. *Integration* of HCI techniques may be extremely difficult to achieve without support from those techniques themselves. If techniques have no explicit design view, or an unrepresentative set of assumptions about the nature of design projects, there is no guarantee that a design team will be able to modify the rest of their approach to suit the idiosyncracies of one specialist approach when they have other issues such as speed, reliability, portability and security to consider. It seems more reasonable that HCI techniques should adapt to design practice, rather than the other way round.

### 7.4.2 Problems with Confidence

HCI techniques seem to fail with respect to convincing designers of their utility in real design practice. The reason for this failure may be to do with a real weakness in HCI, in that it may not be capable of addressing real design, or it may be to do with the fact that HCI techniques are not being presented in a serious form for use in commercial projects. If the latter is the case then HCI techniques need to be more appropriately packaged for systems designers to use, and their developers will have to provide convincing evidence of the techniques' ability to provide real benefits for designers and design host organisations (perhaps in terms of time and cost savings).

The onus is therefore most strongly on HCI DETs to prove themselves, since cynical designers are not likely to attempt to try out these approaches themselves. Researchers need to carry out evaluations of their techniques in use by real systems designers on real design projects. This may be a tall order since such trials could prove highly expensive. Research is required to prove the utility of various techniques applied "in vengeance", and to suggest ways in which they need to be adapted or packaged so that they are more effective and easier to gain benefits from.
7.4.3 Assessing Applicability

HCI DETs may often be inapplicable to systems design projects for a number of reasons. The main implication from this argument is that for any single design project, only a minority of the HCI DETs available, if any, will be applicable in terms of its application requirements and its analytical scope. The problem is how to decide which one(s) are applicable in advance.

There are a number of reasons why one might wish to do this. The designer of an HCI technique might wish to specify exactly what kind of project and user-oriented analyses he or she has in mind for application of the technique. A researcher or company manager might wish to compare HCI techniques with respect to design practice as it occurs in general, or in some special circumstances. A design team might wish to select a particular technique to fulfill some requirements within their design project.

What is required is a framework which provides a coherent view of design practice as it typically occurs and which is able to address relevant issues which may relate to the suitability of an HCI technique for use in real design situations. In the following chapter an attempt is made to outline such a framework. It does not address the design of HCI DETs themselves, nor how they might be improved with respect to application. It simply supports the characterisation of design practice and HCI techniques in such a way that the suitability of one for the other is easier to determine.
8.1 Overview and Introduction

This chapter seeks to provide an overview of the direct evidence on use of user-oriented design techniques from the HCI specialists study reported in Chapter 6. The specialists study suggested that even those with experience in psychology and HCI had problems applying user-oriented design approaches within commercial and applied projects. Two HCI DETs of the type reviewed in Chapter 2 were described by specialists, and three other main user-oriented design or evaluation approaches were identified. Their use and the problems and advantages associated with them suggested a number of desirable features which ideal applicable HCI techniques might possess.

These features are described and combined with other constructs; the scoping matrix introduced in Chapter 2; the design schema from Chapter 5, and the matrix describing support requirements of various roles identified for HCI analysts described in Chapter 6. Together these form a framework for assessing the applicability of HCI DETs. The framework is outlined and then demonstrated in an example of the assessment of the suitability of CCT for project 5 from the systems designers' interview study reported in Chapter 5. This assessment is an expansion of the scoping exercise carried out for CCT and project 5 in Chapter 7, and provides a clearer picture of potential application problems for CCT within the project.

The framework represents an attempt to provide a realistic view of design which HCI techniques should attempt to cater for if they are to be applicable. It does not suggest what scope techniques should have, nor the depth of analysis they should provide. Nor does it provide guidance as to how current techniques should be improved. It simply highlights the obstacles likely to prevent application of un-
compromisingly theoretical and academic approaches in the real world outside the laboratory.

Most systems designers may never be able to develop sufficient expertise in the field of HCI to permit them to determine which technique is most appropriate for their analytical requirements. The assessment framework requires an understanding of the scope and methods of HCI techniques at a level which most systems designers cannot quickly achieve (it requires considerable time and effort to become familiar with even one approach). Therefore it is important to emphasise here that this framework may only be useful for selection of techniques if HCI DET's developers begin to present the scope and all necessary activities required to apply them in an explicit and quickly absorbed fashion. This would be a first step towards encouraging the application of HCI techniques by making selection quicker and easier.

### 8.1.1 Design Practice With or Without HCI

Computer systems and UI design practice (inadequate though it may be) will continue to occur with or without the intervention of HCI tools and techniques. Systems analysis and design methodologies (SADMs) have until very recently, been highly successful without making any but the most cursory concessions to usability analysis from a psychological perspective. This does not mean that design practice benefits from the absence of HCI. Recent initiatives suggest that the developers of some of the most influential SADMs (SSADM, JSD) are now looking towards HCI specialists to find ways of integrating the two fields of systems engineering and HCI (Sutcliffe 1988, Walsh et al 1988). The fact that systems engineers are beginning to recognise a need for a more user-oriented approach is a positive sign for HCI as a discipline, however it should not be taken as a sign that HCI DETs themselves will now be taken up by systems analysts and designers.

HCI will have to adapt to fit in to the constraints which design practice suffers. It is unrealistic to assume that somehow, once designers attitudes towards HCI have become more sympathetic than perhaps they are at present (see Chapter 5), all other obstacles to use of HCI DETs will melt away. Although UI prototyping and simulation tools are beginning to be available, we are still a long way away from fully automated HCI expert systems, or cheap highly generalizable cognitive simulation
tools which designers could apply without having to understand the models driving them. Whilst partial support for UI design is the best we can hope for, the concepts embodied in the HCI techniques systems designers use, and the notations used to express them, will have to be comprehensible to designers in order that they can use them in highly variable, unpredictable design in commercial environments. Furthermore, whether HCI tools and techniques are automated or not, they are still obliged to fit in with the structure of design, and to utilise imperfect input and produce practical output.

8.1.2 Acting upon the Failure of HCI DETs to Penetrate Commercial Design

What appears to be of prime importance, at present, is that HCI researchers come to a better understanding of the process of design which they hope to influence. At present the various design and evaluation methods such as those discussed in Chapter 2 seem to do a reasonable job of capturing properties of UIs which impact on their usability. However, their scope is often limited and has not been proved to balance the effort required to apply them. Little account is given of how to tackle activities which these techniques depend on and the problems likely to arise in a real design situation. Furthermore the techniques have little to say about problems of applying these techniques along with other, possibly conflicting, activities such as architecture design, prototyping, formal specification, porting and so on.

It is unrealistic to expect a single HCI technique to have enormous breadth in the usability issues it addresses, whilst accounting for all design activities on all types of system. However it is necessary to bear in mind the fact that the potential market for HCI techniques is largely comprised of systems designers and analysts who have to consider many more aspects of the design than an HCI DET can satisfy. Therefore an applicable HCI technique should be clear about its limitations as well as its scope, it should also be explicit about what other methods it depends on for input, and what its output represents in terms of its direct impact on design.

There are a wide variety of possible features of HCI techniques which might enable them to be more easily applied, both by HCI specialists and non-specialists. In the following discussions these features will be organised within an application framework which seeks to clarify the nature and relevance of these features. The frame-
work could play a number of roles in furthering the penetration of HCI into applied and commercial systems design practice.

This framework arises from the study of HCI specialists design practice, and observation of the features they found made user-oriented methodologies more or less applicable, and of the features their comments suggested would improve applicability of HCI DETs. The framework also embodies a view of design based upon the findings of two user-oriented design practice studies. This design view sets the context within which the desirable features of DETs should be considered. The framework itself and its design view will be briefly illustrated with respect to two examples in order to demonstrate its power to address issues which arise in real design with respect to existing HCI DETs.

8.2 Implications of the Study of Commercial HCI Practice for Current HCI DETs.

The study of HCI specialists in commercial design raised a number of interesting and highly important issues relevant to HCI which it is appropriate to summarise here. The summary is broken into two parts which reflect the more general findings relating to HCI specialists’ practice, and more specific findings relating to the use and non-use of HCI DETs. To sum up, the study set out to address the following questions.

* What precisely are the HCI specialist’s roles in the commercial design process.

* What design and evaluative techniques do they use, what is the scope of each of these techniques, and how are they used.

* How do HCI specialists using user-oriented techniques avoid or cope with user-oriented design constraints and problems.

* How do user-oriented techniques exploit information sources.

* How do the techniques support the activities of the HCI specialist, and in what ways do they fail to do so.
The following summaries of the findings give a general overview of how the study addressed these questions.

8.2.1 Overview of Findings: General Observations of HCI Practice

The working regimes of HCI specialists' employers were seen to be important in influencing the activities they carried out and the variety of work they undertook. Perhaps the most important issue to emphasise here is that it seems that an organisation can either encourage or obstruct the practice of its HCI specialists, and the fact that HCI specialists are involved in design projects does not guarantee usability of the end product. Late evaluation, isolation of HCI specialists from other design contributors, too few HCI specialists being spread amongst too many projects, and so on may all be dictated by the working regime of the organisation employing the specialists.

Despite the fact that HCI specialists only concentrate on limited aspects of the whole system, a number of roles for them were identified which seemed to cover most of the diverse activities observed. The fact that their activities were so diverse would seem to suggest that an applicable support tool might be more successful if it were to focus on an analyst's role as the organising factor determining which subset of activities it should support.

The study indicated that HCI specialists did not generally have the support tools they required in order to be more effective. Prototyping tools were not generally available, and automated support for the Statecharts used by two of the specialists was not provided. This meant that UI evaluations were usually restricted to almost complete system prototypes in the late stages of design in O2 and O3, and in O1 and O4 the HCI specialists carried out highly painstaking cross-checking of their Statecharts specifications by hand.

HCI specialists were clearly physically separated from systems designers, and there were several complaints about poor communication, lack of understanding and lack of integration. This isolation may have made it even more difficult for HCI specialists to overcome any obstacles imposed by working practices in the organisation. They were dependent on written communication in many cases, but apart from in O1 where integration between different specialisations was observed to be particu-
larly good, no agreed notations apart from informal English were used to communicate between different groups.

Isolation also meant that the ideal timing of the use of HCI and user-oriented techniques was difficult to achieve. Typically HCI specialists would be brought in too late to have effective influence on the design. In O4, for example, specialist E completed a CLG specification only to find that the design had already progressed to the stage that the systems designers were not prepared to make necessary changes to comply with the specification.

Like systems designers HCI specialists were subject to constraints, some of which have just been alluded to above. They felt they needed better resources such as more time, automated support, additional space and people. They often had to work with limited information about users, or only information from non-representative users (such as in-house staff in O3). The complexity and variability of applications meant that any techniques they did use were pushed beyond limits to which they would have been tested, requiring alteration and extension. In addition, TAKD which can be viewed as similar to TAG was found to be poor at capturing the variability of acceptable methods observed in tasks. If used as a competence model of task execution it was seen as unrepresentative, if used as an acceptance model it became too complicated. This is an interesting finding relevant to all HCI approaches, particularly Reisner's Action Language (Reisner 1981) and TAG (Payne & Green 1986), which generate competence grammars. Unless some additional model or empirical analysis determines the most likely as opposed to the ideal methods or all possible methods that system users may select, there may be a trade off between a competence and an acceptance model whereby one is simple but unrepresentative, and the other is representative but complex.

With regard to the exploitation of user-oriented information sources available to them, HCI specialists were able to concentrate on this information, whereas systems designers would have to consider more system and market oriented information as well. It is not surprising therefore that the methods used by the interviewees in this study were strongly focused on obtaining and analysing information about and relevant to users and usability. However, due to the existence of the constraints mentioned above, the information often failed to percolate through to the design of the UI itself. The most commonly used information source seemed to be the experi-
mental subject in evaluation. Presumably organisations feel that people themselves are the best, most flexible and most reliable user simulation technique available. They are probably correct if they do believe this, but experimental evaluations are, at present, generally taking place too late to affect design, and they tend only to highlight faults; rather than suggesting designs.

Perhaps because of the tendency to rely on real people rather than on user models and simulations, late evaluations may be overemphasised in commercial organisations. Real people require real-looking, robust UIs in order that they can simulate user performance. Such UIs may only be available late in the design cycle (Rosson et al 1987). Consequently HCI specialists who rely on user evaluations may be restricted to the role of late-evaluator (or firefighter). In the case of O4 where there was too little space to set up a user-evaluation lab, greater emphasis, than anywhere else in the study, was placed on development of modelling techniques of the type discussed in Chapter 2 with the process of design. In the following sub-section some of the main problems which seem to prevent application of existing HCI DETs will be summarised. By highlighting these problems it should be easier to ascertain what needs to be done to improve future HCI approaches with respect to applicability.

8.2.2 Overview of Findings from Chapter 6: Reasons for Use and Non-Use of HCI DETs

This sub-section briefly seeks to highlight the main problems concerning application of HCI DETs. Since most of the direct information concerning application of these techniques is drawn from the HCI specialist interviews, this sub-section provides a summary of the most important qualities of HCI DETs which enhance or interfere with their applicability to commercial design practice. However, other relevant information was also provided by the two design studies which will also be referred to here.

Reasons for Use of HCI DETs

Since HCI DETs of the type described in Chapter 2 were only found to have been used by the HCI specialists, the reasons for their use have been drawn exclusively from observations based upon the two specialists who did use such techniques
(namely E and F in O4).

**User-centred rather than process- or system-centred.**

The two DETs observed in use in the HCI specialists study were clearly focused upon the user's concerns in the system, rather than the processes in the system. The concerns which govern the spirit of TAKD and CLG are based upon basic psychological theory, such as the existence of multi-layered users' knowledge representations; an assertion common to both of these techniques (Johnson & Diaper 1984; Moran 1981). It was clear that the specialists in O4 who used these techniques did so because they felt these were the most appropriate kind of representation for structuring descriptions which would capture the psychological properties of users.

**Task-oriented rather than functionality-oriented.**

Many SADMs concentrate upon early specifications of the relevant processes which the system will be designed to support (e.g. JSD, Jackson 1983; and SD, Yourdon & Constantine 1978). The meaningfulness of task structures to users may be lost, if they are broken down into functions which either the user or the system may perform (Sutcliffe, 1988, has shown that it is possible to extend the JSD notation to capture complexity of the tasks users would be required to carry out with a system, thus providing a usability metric based upon an early system specification). Both TAKD and CLG may help to preserve task structures, which would have been meaningful to users, within the specification of a system dialogue. The HCI specialists in O4 were able, using these techniques, to produce a structured specification of the users' tasks which, in theory, would be translatable into a design specification for the UI. In the case of CLG, other problems prevented this occurring, however for the specialist using TAKD in a different type of project such a problem did not arise.

**Concrete specification rather than intuitive judgement.**

In O3 the two specialists interviewed tended to base their consultancy judgements on intuition as well as empirical evaluations. In O4 no experimental evaluation facilities were available, and the two specialists were anxious that their contribution to design should be as scientifically rigorous as possible. The DETs which they
used were therefore chosen as methods which had been carefully developed with the aim of permitting a psychologically realistic structure to be reflected in an explicit analysis of tasks. The completeness of these analytical specifications helped to ensure consistency and validity of the actions and objects involved in task descriptions which could then be reflected in the design of the UI and its dialogue.

*Design driving or supporting rather than evaluative.*

In 04 the lack of experimental facilities meant that the two specialists tended to channel their efforts more into influencing design rather than evaluation. The techniques which they chose to assist them in this were therefore ones which allowed them to produce specifications of to-be-supported tasks which were essentially independent of the actual design specification itself. In the case of CLG the specification happened to be based upon an existing version of the UI to the application in question, however it could have just as easily been based upon unsupported tasks. TAKD and CLG seemed to be applicable relatively early in the design projects described by specialists E and F. This may have been an important feature governing their selection.

**Reasons for Non-Use**

The reasons given below for the non-use, or limited use of HCI DETs are drawn from all of the studies reported in this thesis, and represent a very general summary of some of the findings.

*Lack of automated support.*

Complaints about the time consuming nature of HCI DETs which arose in the Designer Interviews study and the HCI Specialists study tend to support the view that the complexity of the specifications required for real systems is generally prohibitive of their application. Furthermore the reliance on psychological expertise noted in Chapters 3 and 6, in particular, means that even greater effort must be required of the non-HCI specialist. HCI specialists A and F who both used Statecharts (Harel 1987) were keen to acquire support tools they knew to be available for this technique (STATEMATE Ad-Cad 1986; Statemaster Wellner 1989). Although they were clearly satisfied with the unsupported notation, they found it highly time con-
suming, and requiring of painstaking error checking. No HCI DET is currently supported to the extent that commercial use could be made of the support tool, although prototypical expert systems do exist (e.g. for ICS; Barnard et al 1986). It would appear that the problem of making such tools comprehensible to non-HCI specialists is greater than that of supporting or automating the notations or the structure of the models produced.

_Lack of general comprehensibility._

In itself the lack of general comprehensibility seems to be a major problem for HCI techniques. Since psychology is not a common subsidiary course for computer science degrees, and was probably never taught on early computer science degree courses, it is hardly surprising that there seems to be a culture gap between HCI specialists (many of whom have little programming and system design experience; see Chapter 6) and systems analysts and designers (who often have little or no HCI experience; see Chapter 4). Not only are the notations used by HCI DETs frequently idiosyncratic and visually complex, but the very concepts which they attempt to convey are highly esoteric, and poorly related to ideas and terminology likely to be helpful in application of the specification to design. HCI specialist E in O4 provided the most striking evidence for this feature of HCI DETs when he described the response he got from a CLG specification he passed on to the systems engineers. They found the specification to be too complex to understand and did not make use of it.

In their use of user-oriented DETs HCI specialists in O1 and O4 were clearly reliant upon their skill and initiative in carrying out appropriate additional activities not guided by the technique itself, such as task analyses, dialogue specifications, and system scoping. They also found that their experience was essential in maintaining usability principles with Statecharts which does not explicitly support usability. Specialist E himself found CLG extremely difficult to use even with his experience, and E and F found it necessary to make alterations and extensions to TAKD to suit their particular requirements.
Lack of integration.

A major problem for HCI specialists seems to be their physical isolation from the rest of the design team. Evidence from the design questionnaire study, and from the interviews with HCI specialists tends to suggest that their work may easily be ignored by the majority involved in the project who may not recognise the value of HCI, and are therefore not prepared to make a special effort to incorporate the advice of HCI specialists into the design.

This problem cannot be helped by the fact that the methods presented in HCI DETs do not relate clearly to the rest of a system's design. Many HCI DETs seem to assume that the UI or just its dialogue can be completely defined without integration with, involving input from, and output to the rest of the design process. This may, in theory, not be a problem if one assumes that the activities external to the HCI technique will be properly carried out in such a way that valid input is provided to the technique, and that its output is taken up in the software specification. However the targets of the HCI specification may not apparently be related to the targets of a system specification.

For example simple tasks (as embodied in TAG; Payne & Green 1986) are not elucidated in SADMs (for example JSD; Jackson 1983; has no comparable notion). It seems to be up to the systems engineer, who will probably have difficulty appreciating a psychologically founded notion, to take up a specification couched in such terms and translate it into something more directly related to software related notions.

A further problem for integration of HCI DETs is that of the timing of specification, evaluation and respecification. Reisner's Action Language (Reisner 1981), TAG, GOMS, and CCT seem to require a well specified dialogue. in the case of TAG a high value is attributed to the ability of the technique to capture real world knowledge, and to highlight different kinds of ‘consistency’ (consistency meaning a number of different properties according to the authors of TAG). However the level of detail to which a dialogue has to be specified to reveal the superiority of one UI over another makes it highly unlikely that the design is likely to be highly modifiable on the basis of an HCI evaluation of this type. It seems even more unlikely that the superiority of TAG over Reisner's Action Language could become apparent long before the UI design was complete. This suggests that such tech-
niques may not, in practice, yield results of any great value to a design team since the information they provide cannot be acted upon (unless the UI is very highly modifiable even at the end of the design process).

**Lack of resources.**

Since lack of resources, including time and technology, seems to be a pervasive problem for all involved in systems design, it is unlikely that the kind of resources required for a GOMS or CCT analysis would be provided by any but the largest of projects. Cheap, easy-to-use support tools would ameliorate this problem, but the effort required for a CCT specification (i.e. programming in a large number of production rules to the simulation of the user) would still necessarily take a great deal of human time. The systems designers interview study revealed that HCI is seen to be too time consuming and expensive to be worth while. In other words, as has been stated previously (see Chapter 7; sub-section 7.1.4) the benefits of application of a technique must be shown to outweigh the costs. This is a property which has not been proved for any of the techniques described in Chapter 2, and needs to be considered carefully given the existing mistrust of HCI DETs which was revealed most clearly in Chapter 5.

### 8.3 Ideal Properties Required of Applicable HCI Techniques.

In this thesis much has been said about the shortcomings of HCI DETs with respect to the needs of systems designers and HCI specialists in commercial practice. The unsatisfied application requirements discussed in Chapter 7, and the findings from chapters 4 to 6 illustrate the need for greater attention to be paid to applicability if HCI is to avoid restricting itself to descriptions of existing technology. Throughout the greatest part of this thesis, the emphasis has been on properties of HCI techniques relating to application, with an implicit acceptance of many of the assertions and claims made by the authors of the techniques reviewed with respect to their psychological validity and their descriptive, explanatory, or predictive powers. Whether the theoretical aspects of these techniques are assailable or not seems to be of indifferent importance as long as HCI fails to gain an accepted place in UI design practice.

An important implication of the research discussed in this thesis is therefore the
necessity for HCI, as a discipline, of viewing design practice and its constraints in much the same way as HCI DETs recommend that systems designers should view the user. One way to convey this point is to frame it in terms of a metaphor in which the system life-cycle and the people involved in it are equivalent to the USER, and the HCI technique is equivalent to THE SYSTEM and its UI. The designer of the UI, according to HCI advocates, should attempt to design the UI to suit the needs of the user. In order to do this the designer has to consider what qualities the UI should embody which will make it more suitable to the requirements of the user. In the same way, designers of HCI DETs should consider what types of properties their techniques should embody in order that those involved in the system life-cycle will find them easy to apply. Such properties would be roughly equivalent, in terms of this metaphor, to the usability principles introduced in Chapter 1.

In the following discussion a number of types of properties are discussed which have been drawn from the three studies described in chapters 4 to 6. Some of these are essentially those which were popular in the existing user-oriented methodologies applied, others may be more idealistic, based upon observation of problems and constraints which they might help to overcome. By adhering to these properties as though they were principles, HCI DETs may significantly reduce their application requirements (see Chapter 7).

8.3.1 Limited Investment or Major Pay-Offs

It seems obvious to state that a technique should pay for itself if it is to be commercially viable, however this type of property has never been proved for an HCI DET. Mantei and Teory (1988) attempt to show that, in principle at least, a user-oriented approach to UI design can save money. However the evidence for HCI DETs being commercially viable is scant, or non-existent. Sharratt’s study of students attempting to use CLG (Sharratt 1987) suggested that the students spent more time worrying about their usage of the CLG specification than they did on the actual design. The HCI interview study also showed that an attempt by specialist E to apply CLG in practice turned out to be largely a waste of time.

The costs of applying HCI techniques are likely to be in terms of requirements for skilled individuals, extra time required to gather the necessary user-oriented infor-
mation and analyse it, tools perhaps for supporting techniques or building user or system simulations, and effort required to make changes in response to any evaluations.

The benefits of applying a technique are likely to be dependent on certain combinations of properties of that technique such as the stage in the design process at which the output from that technique is available, or the importance of the aspects of the UI which it addresses to the overall usability of the system. The property of being able to drive user-oriented design would be more valuable in terms of benefits for that design than the ability to evaluate it, given a fixed amount of effort. If a technique can explain UI problems rather than just highlight them it is likely to be more valuable.

This property is really dependent upon the existence of the other following properties within a technique. If they are indeed apparent, then it seems more likely that a technique which embodies them will be of some value to UI designers. As stated previously, the onus is on the developers of HCI DETs to prove that they are worth applying. It is not within the scope of this thesis to suggest for each technique how best it might be made "cost-efficient". However, for each of the following properties, ways of determining their presence in a technique are suggested. If developers of techniques recognise the absence of valuable properties in their approaches, and desire that their techniques be used in applied design, then they may wish to address those properties by modifying or repackaging their technique in some way.

8.3.2 Accuracy

As stated earlier, the accuracy or validity of various HCI DETs as claimed by their authors, is not called into question here, although systems designers seem to be sceptical about them (see Chapter 5). Various authors (e.g. Card et al 1983; Kieras & Polson 1985; Payne & Green 1986) claim that their techniques produce predictions about user behaviour which are accurate to some degree when compared against actual user performance. However the degree to which commercial systems designers or HCI specialists can replicate the laboratory results in the field is highly contestable. There are two main reasons for this, the first being the difference in experience and skill in the technique between the author and the practising analyst, and the second being the difference between the contrived laboratory circumstances
and the real world where design takes place.

It is important that the developers of HCI DETs provide convincing demonstrations under realistic design conditions which show that the technique is appropriate, does not require an inordinate amount of difficult extra work, uses input which is normally available, and provides output at a time when it can be profitably exploited, before the system software becomes unmodifiable. Such measures are necessary in order to persuade skeptical systems designers that a technique will produce useful results for them, since they seem to be well aware of the difference between theory and practice. The "this is the real world" philosophy revealed by the design studies in chapters 4 and 5 may be hard to dislodge without convincing evidence that theoretically based HCI DETs can produce accurate and valuable information and predictions.

8.3.3 Expressiveness, Clarity and Communicability

The notations and constructs chosen by HCI DETs seem to vary widely in form and expressiveness from the formally precise BNF to the boxes and connecting arcs in BIMs. Each notation seems to have been chosen with the intention of capturing the information of interest in a form which enables the analyst to draw the appropriate inferences, and, in the case of the more highly structured notations, make precise statements about various properties of the UI, dialogue or user's representation. The feature of expressiveness has little to do with validity (i.e. the probability that the notation represents something real in the user's head) and more to do with descriptive power. Hence, if we assume that system users really do construct hierarchies of interaction rules which are not related to or influenced by any semantics, and other real world knowledge, then we may have no argument with the form of Reisner's Action Language which is clearly highly expressive. However if we assume that psychological validity is important, and our notation, like BNF has difficulty in capturing notions such as semantic restrictions (Reisner 1981) and real world knowledge (Payne & Green 1986) then our notation is not expressive enough to deal with these properties of users. In this sense we may assume that TAG is a more expressive notation for representing users knowledge than the Action Language is. Expressiveness is an important quality for an HCI DET since anything which cannot be expressed will either have to be ignored or added in an impromptu fashion by the analyst, with the subsequent risk of defeating or interfering with the
spirit of the technique.

Certain constructs, most notably the MHP cognitive processing architecture (Card et al 1983) and the ACT* architecture (Anderson 1985) may be added to express the assumed performance properties and constraints of system users. In the case of competence models such as the Action Language and TAG such constructs are merely implicit, and as such it is more difficult to invalidate the models they produce, as well as to be certain of all their implications. The more a technique relies on implicit assumptions, the less likely it is that an analyst, using the technique for the first time, will be able to use it as intended by the inventor of the technique. The more expressive a notation and constructs associated with a technique, the more likely it is that the results obtained from its use will be consistent between analysts.

In practice the clarity of the notation and representations of the constructs involved in a technique are probably as important as the expressiveness. CLG is claimed to be capable of expressing precisely the rules for generating interaction sequences from the syntactic level (Moran 1981). However its lack of a clear notation makes it difficult to interpret without a good deal of scrutiny. The Statecharts notation may be more visually appealing because it capitalises on human perceptual and information processing strengths (see Chapter 6).

Expressiveness and clarity when combined should help to ensure communicability of an HCI DET. If HCI techniques are designed to be used by both HCI specialists and non-specialists in applied design projects, they should be more appealing to systems designers and more easily transmittable to various members collaborating on a project. Perhaps two important questions which the authors of HCI techniques to be used by non-HCI specialists should be asking themselves are: "Is it clear, in non-esoteric terms, what exactly this technique entails, including the activities it depends on?" and "Would non-specialists be able to do this?" In other words techniques should exploit notations, constructs and terminology which are unambiguous and comprehensible to the analyst producing the specification and the recipient of a specification. Representation is not the same as communication; a notation may claim to capture properties of what it represents, and the initiated may be able to interpret it. However, for the HCI specialist to be saved the onerous task of rewriting a model/specification or part of it in order to communicate it to others, the specification should be expressive, explicit and clear.
8.3.4 High Generality and Broad Scope

A major problem with HCI DETs at present is that they suffer from a trade off between breadth and depth of the properties of the UI or the user which they are able to capture (Barnard 1987). For example BIMs cover a wide variety of knowledge sources and indicate how they might interact, however they are not detailed enough to produce any clear predictions about the effects of interacting knowledge sources on user performance. GOMS, on the other hand, produces performance predictions about speed of task completion, but will not account for behaviour of anything but an expert system user who may be fast or slow at the task (Card et al 1983). HCI DETs seem to cope with only a narrow range of behavioural properties of users, but within these restrictions, they tend to produce highly detailed models and predictions.

HCI DETs may also be generalizable to only a small number of systems; for instance text-editors, electronic mail systems or drawing systems. The questionnaire design practice study revealed that many systems will not fit into these categories, yet HCI DETs seem to focus on such systems and text editors in particular. In order to be generalizable to a wide variety of systems a model must be able to account for a wide variety of user behaviour types, such as perceptual processing, problem solving, learning and so on. Many important types of interaction or computer supported work, such as direct manipulation, diagnosis from digitised images (as in geological or medical work), statistical modelling, and design (of clothes, buildings or aeroplanes, for example) will not be adequately described by models which only address limited behavioural properties of users. This leads us back to the previous point which was that breadth of HCI DETs is typically limited to a small number of user characteristics, such as representation of dialogue syntax (as in Reisner's action Language and TAG) and representation of routine-task goal stacks (as in GOMS and CCT). There seems to be a need to address more diverse types of user behaviour in order to improve the applicability of HCI DETs for a wider range of types of system. In other words, the scoping matrix, introduced in Chapter 2, needs to be expanded in the evaluation factor row for users into a more diverse set of rows which will allow it to show more clearly the range of user behaviours an HCI DET is capable of addressing.

It is also worth remarking on the problems of dealing with user variability which
make models more complex, because they have to include a variety of methods for accomplishing goals (an acceptance model of interaction) or risk becoming unrepresentative. As has been shown by Young and Maclean (1988) it is possible to reduce the problem of unrepresentativeness by adding a predictive model of how users select one method over another in a particular set of circumstances.

8.3.5 Successful Exploitation of Information

Exploitation of the limited available user-oriented information in a design project may be crucial to the success of the UI. Prototyping is becoming an increasingly popular form of system development (see Chapter 4), and seems to represent a potentially valuable source of information. If it is impossible to get access to representative users can HCI DETs help us to extrapolate from the behaviour of unrepresentative experimental subjects, or in the worst cases, from the behaviour of members of the design team using a prototype? GOMS and CCT expect designers to empirically derive basic operator times as input to their models. However they do not encourage designers to simulate other aspects of behaviour such as error making and recovery, inefficient method selection (selection rules in GOMS are based on competence rather than incompetent performance which is equally likely), pauses for thought (when might these occur; a prototype might highlight difficulties users need to think about), and so on. As far as designers may be concerned, prototyping using non-representative experimental subjects may be more predictive of actual user behaviour than a GOMS or CCT simulation. Systems designers need to be shown that HCI simulation techniques can provide more valuable information in such circumstances, if indeed they can.

Techniques which depend on task analyses need to maintain all of the necessary information which would enable the UI designer to provide appropriate support for the user. In the HCI specialist interviews it was found that TAKD tended to lose so much valuable contextual information that only the application domain expert who provided the task information, and the analysts themselves could follow the analysed generic task representations. This made it necessary for designers to have the tasks they needed to support explained to them. In a design project where task analysts probably work separately from systems designers (see Chapter 6), contextually rich communication of user-oriented information is necessary to ensure that the value of task analysis is not diminished.
8.3.6 Integratability

Integration of HCI, given the current organisational restrictions which separate HCI specialists from others involved in design, may rely on a number of properties of HCI DETs. The input to a technique, its output and the point in the system lifecycle at which these are produced are all important issues which have to be addressed by applicable HCI techniques. Many depend upon activities such as task analysis, system models and user models which provide the input from which they derive descriptions or predictions. It is not safe to assume that the relevant information will be available early enough for an HCI analysis to produce influential results.

If for example a TAG analysis were chosen for a UI, but the design project was working along the lines of a top-down design approach (as CLG does), the predictions about usability would only become available at the end of the project when the details of interaction were being established. The design team would have to have ensured that code was easily modified right up to the last moment. As the designer interview study revealed, this can be a problem (see Chapter 5, project 3). It is not always possible to alter code late on. Perhaps the only solution to this problem is to highlight the need for ensuring modifiability of code throughout the design process, as object-oriented approaches attempt to do (Jacobsen 1987; Meyer 1988). On the other hand, tools such as CLG with a strong top-down bias, would probably not work well with more bottom-up approaches (such as object-oriented design). So it seems that an important feature of an applicable HCI DET is a realistic or at least explicit view of the design process within which it might be used to best advantage. However, most HCI DETs are not even clear what type of model they are at present let alone how they might be used (Whitefield 1987).

Integration is also dependent on shared notations, terminology and concepts, in other words the potential for communication has to be improved between the HCI and systems engineering disciplines. And finally the roles of those working within within a design project must be considered as discussed in the following subsection.
8.3.7 Support for Analyst

Although this type of feature has been left until last, it may be one of the most important and easiest to satisfy. By support, we should not restrict ourselves to the notion of computer support or automation. Support can be given in the form of extra guidance, or a more coherent approach grouping of activities which the HCI DET can assist. CCT, for example, is presented as an early evaluation tool (although this claim is contested elsewhere in this thesis) and as a possible teaching tool (GTN system UI representations are supposed to help users to construct a more efficient model of the system's behaviour; Kieras & Polson 1985). No coherent role in which these two types of activity might be combined is presented.

The HCI specialists interviews revealed a number of HCI practitioner roles which techniques might support (see Chapter 6). This would mean considering all of the activities important to each role and placing an emphasis on ensuring that these are given special attention, for example; techniques which are intended for technology transfer exercises must be easy to demonstrate, flexible enough to suit a variety of types of project, and explicit as to all of the necessary activities, and the design approach which they depend upon, failure to ensure these features could lead to the kind of problems which arose in the technology transfer project described by specialist B in O2 (such as over general or inapplicable guidelines; see Chapter 6).

Analysts need guidance for the activities upon which a technique depends, which are often not clarified by the technique. Two HCI specialists in the interview study of HCI practice, complained that TAKD and CLG required certain activities to be carried out without explaining how one would do so appropriately. Non-HCI specialists would need further guidance because they lack the psychological expertise necessary to perceive implications from analyses, and may even fail to understand the terminology and concepts involved in the first place. In other words, since HCI specialists are a rarity in commercial design projects (see Chapter 4) an applicable HCI DET should have limited dependency on skills and intuition, and needs to come with more guidance than seems to be the case at present.

The compactness or economy of the components of a technique (its notation, the user characteristics or the system properties it addresses) may have a good deal to do with its comprehensibility and the amount of automated support it actually re-
quires. It would appear that HCI DETs can become tedious or highly complex to apply as reported in Chapter 6, and even specialists may become discouraged by this. In addition the amount of time and effort required may be very large for non-trivial UIs. However, broader, psychologically valid and detailed models will naturally be complex since the users which they aim to simulate are highly variable, complex, context sensitive systems involving perceptual processing, attentional variations, memory constraints, diverse experience and roles and so on. Complicated models which can capture multiple properties of users are highly likely to require computer support for the analysts who use them, perhaps embodied in expert systems or knowledge engineering or simulation tools. So far such support seems to be limited, or non-existent.

This informal classification of desirable properties of HCI techniques needs to be refined into a number of more concrete features which an HCI DET should possess. In the following section a number of specific features are drawn from these more general properties to stand as a checklist for applicable HCI DETs. Their specificity should make it easier to determine whether or not they are indeed embodied in a given technique.

8.3.8 Some General Points

The types of feature addressed in the discussion above raise some interesting general points about applicability of HCI which are summed up below. This may seem like stating the obvious, but as Gould and Lewis (1985) showed with their so called obvious design principles; the obvious may not be as well recognised as it at first seems.

* Rather than aiming for ever more refined and sophisticated models, techniques should perhaps be aiming for more complete and coherent support for their application, in terms of realistic views of design and activities they depend upon, and roles which they might support.

* HCI DETs must aim to embody properties that make currently applied practices popular (irrespective of HCI aspects they may or may not have). For example the Statecharts reactive-system representations technique was popular because of its clear representation, simplicity and generalizability.
8.4 A Classification of Specific Desirable Features of HCI DETs

This and the preceding chapter have summed up a rather negative view of the impact which HCI techniques seem to be having on applied and commercial design practice. The discussions do not reflect the validity of user modelling efforts, nor the ability of theoretically based techniques to produce representative predictions and explanations of interactive behaviour. The real problem seems to be one of packaging HCI for use by non-specialists, or for specialists operating in constrained applied design projects, as opposed to contrived laboratory studies.

In this section the findings which emerged from the investigations reported in this thesis are brought together and used to construct a list of desirable features for assessing future HCI DETs with respect to applicability.

The sub-sections 8.3.1 to 8.3.7 represent an informal classification of ideal properties of HCI DETs. In another sense it is possible to say that each type of property is really a basis for evaluation of the applicability of a technique. Examining the degree to which a technique appears to embody a particular type of property is rather like evaluating it from a particular perspective. However the above classification, which is based upon the interviews with HCI specialists in commercial organisations, is too unclear, since some of the types of property seem to overlap with others, for example Limited Investment or Major Pay-Offs overlaps with Generality since both would demand that a technique should cover as wide a scope of user behaviours as possible. It is therefore necessary to make more rigorous demands of HCI DETs in order that they might be considered more applicable.

In the following, each of the property types (in italics) will be broken down into more specific, less overlapping features (in bold type) which will be referred to from here on as Desirable Features of applicable HCI DETs. Each specific feature will be defined with the intention that it should be possible to ascertain whether or not it is present in an HCI DET. The responsibility for proving each feature's presence lies with whoever is most determined to see a technique applied successfully; in most cases one might assume that this should be the responsibility of the creator(s)
of the technique.

The possible consequences of the embodiment or lack of each of these features will be outlined with respect to evidence from the HCI specialists interviews from which these desirable features are drawn (see Chapter 6; findings of the study). Of course this is not to say that the feature was either fully or not at all addressed by the technique concerned in the example, it merely illustrates the consequences of adequacy or inadequacy in some aspect of the technique with respect to the feature in question.

Limited Investment or Major Pay-Offs for an HCI DET can be viewed as a trade off between the amount of effort required; i.e. simplicity of the analysis, and the scope of the technique. Most techniques are fairly clear about their scope, however little is said about their simplicity. If systems designers are going to be able to make an informed decision about which technique they can afford to use then they need some information about the time and expense they will have to invest (see Chapter 7), which can be balanced against the benefits of the technique. However, we should not combine these two features, since the importance of each will vary. If the expense is not important then one might wish to focus only on how great the scope is, and vice versa.

1. Simplicity
   Defined as the number of constructs and different activities required by a technique, which would have to be learnt by an analyst in order to use it. Obviously the more esoteric a technique is, the more its use will be restricted to HCI and psychology specialists.

   Evaluated experimentally and judged either in isolation or compared with a more common design approach as Boehm et al (1984) have done with prototyping versus specification (see Chapter 3). It could also be evaluated self referentially by any HCI DET which claims to capture the property of complexity in a representation of a task execution.

   Presence of the Feature of simplicity of a technique was provided by the use of Statecharts which were described by two HCI specialists as being easy to learn and use. The main problem was the size of specifications which they felt they needed
Automated support for.

Absence of the Feature was demonstrated by the use of CLG which was reported to be difficult to learn, use and understand by an HCI specialist. It involved understanding of the nature of each level of description, and of the syntax of the notation, and the rules for translating one level to another. Furthermore it produced a very lengthy specification which systems designers could not understand.

2. Broad Scope

Defined as the breadth of properties of a UI to which a technique is sensitive, and the breadth of properties of a number of factors (users, application, UI, target tasks, and performance) which it accounts for.

Evaluated by ascertaining how many different principles (i.e. properties which one might strive to achieve) of usability (e.g. simplicity, compatibility) can be captured by the model which the technique generates, and the properties of the evaluation factors addressed. In other words, the scoping matrix used to characterise each of the techniques reviewed in Chapter 2 could be expanded and applied to a technique to determine how great its breadth was. In addition it is possible to expand the contents of each cell to give either a weighting of the degree of rigour or accuracy with which it is represented, or a description of how it is tackled by a technique.

Presence of the Feature of broad scope was demonstrated by the scoping of experimental evaluations (see table 6.5) which, by their nature, can simulate an extremely wide number of properties of the factors essential to evaluation (i.e. users, applications etc), and reflect the influence of the presence or absence of system properties (i.e. the various principles) on human behaviour.

Absence of the Feature was illustrated to some degree by TAKD (see table 6.8) which, although it worked quite well for its analysts, had to be extended to address allocation of tasks (to the system and to the user which would have addressed the application factor) and task hierarchies to show relations between tasks and task dependencies (which would have given a more rich picture of the simplicity of the overall task descriptions).

Accuracy of predictions as claimed by the authors of a technique, when it is applied
by another analyst, will depend upon the explicitness of the entities and procedures it entails. In much the same way as an experimental report must be written clearly enough to allow results to be replicated, so the definitions of the components (e.g. simple tasks and cognitive processing constraints) and the methods which involve building a model from them must be laid out in an absolutely unambiguous manner.

3. Explicit Entities and Procedures
Defined as the extent to which a technique is self explanatory, and can be used by a non-specialist without prior knowledge being assumed in the description of how to apply it.

May be evaluated in much the same way as one would try to run a new computer program, by trying it out on a "naive user" in order to determine if he or she can complete analyses and obtain valid results with it.

Presence of the Feature was a main aim in the design of the technology transfer technique used by O2 in the HCI specialists study. The technique avoided jargon, and specialist knowledge. Most of the methodology involved filling in forms, charts or matrices, and the technique involved workshops in which designers were taken through the technique step-by-step.

Absence of the Feature was fairly apparent in CLG and TAKD in that they both explicitly rely upon collection and analysis of task-oriented information which the analysts using them conducted, using their own skills and experience, but which the techniques themselves do not explicitly guide.

Expressiveness, Clarity and Communicability. Communicability is assumed to depend upon the expressiveness and clarity of an HCI DET. Expressiveness depends upon whether all of the components which affect predictions of a technique can be expressed in terms of the notation and constructs involved. For example it is not appropriate to assume that all analysts will realise that complexity of a UI will be less of a problem for a user if much of that complexity is already real world knowledge that the user will have. In such a case the notation would have to include some way of capturing real world knowledge and indicating that it would not count towards a complexity metric; for example, this has been achieved to some de-
gree in TAG (Payne & Green 1986).

Clarity may be dependent upon the visual simplicity of the representations or notations used by the technique, and upon the degree to which the semantics of the technique are conveyed by the symbols or terms involved (this is covered by the explicitness feature).

4. Expressiveness
Defined as the degree to which the theoretical underpinnings and concepts (such as cognitive architectures, human memory properties and so on) are reflected in the explicit components of the technique.

Evaluated by identifying each claim made by a technique as to the properties of a user or system which it is sensitive to or claims to be able to model, and then checking that a complete specification using that technique represents each of those properties directly.

Presence of the Feature was apparent in Statecharts which makes no claims beyond the properties which it can be formally demonstrated to possess (Harel 1987). However in the context within which it was used by HCI specialists, user-orientedness of the specifications had to be preserved by keeping in mind properties which might affect usability which are not explicitly encouraged by the Statecharts technique.

Absence of the Feature was demonstrated by CLG which did not directly represent any of the user's psychological characteristics which would enable a psychology naive analyst to determine whether or not a system was usable. Fortunately, the specialist applying it had enough experience in psychology to compensate for this lack.

5. Visual Simplicity
Defined as the ease with which the notation used by the technique can be deciphered, both by experts and naive analysts.

Evaluated either experimentally by asking subjects to attempt to translate a specification into some other representation without worrying about the semantics
(perhaps a flow chart, or English) and analysing what problems they had. For example parts of a CLG specification could be represented in terms of an execution diagram. Visual complexity may also be evaluated by applying some model of the visual processing system which is able to predict what are the causes of difficulty of interpretation of visually presented information (e.g. Kosslyn, 1989, presents a model based upon Gestalt and other processing theories of visual perception which he claims may be used to evaluate the complexity of graphical information). It may be sufficient to analyse a textually complex specification using some notation in terms of the violations of the type of recommendations made by visual processing researchers (e.g. Waller, 1980; Wright 1980) which suggest use of lay out, font, spacing, punctuation and so on can all improve clarity of text.

**Presence of the Feature** was described by both of the specialists using Statecharts who claimed that the charts were easy to read and comprehend, and could be used to communicate to programmers who could write software from them.

**Absence of the Feature** was evident in CLG where a secretary made many mistakes in copy typing the notation from the specialist's hand written notes, and systems designers were unable to decipher the specification.

**High Generality and Broad Scope** Broad behavioural scope here particularly focuses upon the range of user behaviours which a model is able to capture. **High Generality** is intended to imply the ability of a technique to address issues related to a wide range of UIs, applications and target tasks (see Chapter 1). These two are possibly two features which may be traded off against one another. On the other hand it may be the case that after a certain point, increasing the scope of a model employed in an HCI DET *improves* its generality because it contains more cognitive components which may allow it to explain behaviours with a wider range of UIs.

6. **Broad Behavioural Scope**

Defined as the range of human behavioural (including cognitive) properties which are captured or addressed by a technique.

May be evaluated by referring to the HCI scoping matrix (see Chapters 1 and 2) and expanding the row which relates to the user as an evaluation factor. Candidate *sub-row headings* might include items in the following list, (general mental properties
such as short and long term memories, which underlie common assumptions for all psychological interpretations of behaviour, are not included as useful behavioural candidates which would serve to distinguish between HCI models).

Learning
Perceptual Processing
Knowledge Representation
Language
Planning
Problem Solving
Proceduralised Behaviour
Errors
Social Behaviour
Organisational Behaviour

Each of these classes of human behaviour could then be mapped to the usability principles in the scoping matrix and a user or interaction model could be characterised in terms of the cognitive behavioural properties it was attempting to capture, as influenced by the properties or principles of interest. These were not represented in the original scoping matrices used on the HCI DETs described in Chapter 2 since those techniques typically concentrate on few of these properties; predominantly; knowledge representation, language and proceduralised behaviour (with some predictions of where user errors might occur in tasks). Whether or not the list of behavioural classifications represent a complete set of all the behavioural aspects of system users which could be important to HCI is not at issue here. What is of interest is whether these classes help us to characterise an HCI DET more clearly for the purpose of evaluating its applicability.

Presence of the Feature could have been apparent in experimental evaluations had the analysts been able to design their experiments such that they were sensitive to these properties. However the study did not pursue this aspect of the use of experimental evaluations. It may be that the design of commercial user-evaluative studies focuses on the UI causes of problems, rather than the interplay of cognitive processing components leading to slow performance or mistakes. The evaluative nature of these experiments may require only that bad design is highlighted, and not that the psychological reasons for its being bad are described and explained.

Absence of the Feature was seen in the technology transfer technique which only
focused upon the tasks and roles of system users. Much of the scope of this technique may lie outside the scope of the matrix, being concerned with markets, businesses and so forth.

7. High Generality

Defined as the extent to which variations in the evaluation factors in the scoping matrix can be handled by the technique. A particular technique may deal with all the factors for one design; say trained users, text editing application, command driven UI dialogue, simple editing tasks, and performance metrics with respect to these, and yet may be inadequate for another involving naïve users; air traffic control training applications, menu and direct manipulation UI, monitoring and complex decision making tasks; and appropriate performance metrics with respect to these factors.

May be evaluated by taking a diverse range of UI, application, and task properties, selecting representative combinations of these properties and demonstrating an HCI DET's ability to capture and explain problems or interesting occurrences during interaction with a system with these properties.

Presence of the Feature was once again apparent in experimental evaluations which appeared to be possible for any system which had a robust enough UI for user testing to take place.

Absence of the Feature was described by the specialist reporting the technology transfer technique who stated that some systems designers trying to apply the technique in practice found it redundant in that many of the activities it required were not appropriate for certain applications.

Successful Exploitation of Information relates to how realistic certain application requirements of an HCI DET are given the likely information available in an applied or commercial design situation. If an HCI DET does not exploit information which is readily available, but demands information which is not, then it is unlikely to be applicable. Two specific features are important to this feature type, the realism of the design view an HCI DET embodies either explicitly or implicitly, and the extent to which information input to the technique is preserved in the output. HCI DETs usually state what information they are trying to convey in an analysis. Any
information loss could be wasteful and could prove problematic for recipients of a specification based upon a technique, if they have no knowledge about the original input data.

8. Realism of Design View

Defined as the extent to which the design process assumed or envisioned by a technique reflects realities of situations where it claims it should be applicable. These situations may be design practice in general, or a particular type of situation or approach.

Evaluated by comparing the statements relating to the input for and output from a technique, plus any pragmatic application requirements, with conditions existing either in a target, complete SADM or in a representation of commonly occurring design practice such as presented in Chapter 5.

Presence of the Feature seemed to be apparent in TAKD which, as a task analysis technique used input and produced output which was available, and produced output which was useful for the projects it was used in. It did not make demands for the existence of system specifications or empirical performance data which would not have been available.

Absence of the Feature Showed up strongly in CLG which had a top-down approach but a separate starting point from that of the software designers on the project it was used in. Since the analyst who used CLG began his specification at the same time as the systems engineers began writing the software, the length of time the specification took reduced its chances of being applicable. By the time the interaction level had been specified, and could be communicated to the systems engineers, they were unable to alter their software to suit the specification. CLG is not presented with any warning as to the possibility that this might happen, or how to avoid or deal with it.

9. Preservation of Important Information

Can be evaluated by testing the ability of a complete HCI DETs model to convey the required design supporting information to a systems designer who has no knowledge of the analysis which provided the original input data. If additional explanation is required to enable the designer to produce a design or problem solutions
from the specification then important information may have been lost. In such a case that information may only be implicit in the analyst's head, and only the analyst could produce a design based upon the output of the HCI DET.

**Presence of the Feature** was described by the analyst using CLG who stated that the technique was useful for maintaining a coherence and task-orientation of the specification through to the most detailed level.

**Absence of the Feature** was notable in TAKD in that the analysts who used it stated that it lost so much contextual information during the process of abstraction that it required high familiarity with the analysed tasks themselves in order to understand the task specifications. This proved to be a major problem when attempting to communicate the TAKD specification to software engineers.

**Integratability** assumes that an HCI DET should, without major modifications, or extensions, fit into a typical design cycle, that it should not depend upon activities which are typically not carried out within the type of design cycle it is applied in, and also that its output will be appropriate as input to other design activities. Essentially the integratability of an HCI DET would have to be determined by the Realism of Design View (see above).

**Support for Analyst** would appear to be dependent upon two specific features; the degree to which it restricts its methods to Explicit Entities and Procedures (see above), and the degree to which it is able to cover a coherent set of activities which would aid the analyst across the whole of a project. This would mean having a clear picture of a possible role for the analyst and supporting each of the activities which that entails (five HCI oriented role types were identified in the study of HCI specialists, including that of "designer"). We can refer to this feature as the coherence of support which an HCI DET provides.

10. **Coherence of Support**

Defined as the completeness with which the activities within some HCI analyst's role are supported by a single technique.

Evaluated by determining the most likely role of an analyst using a particular technique (given the explicit aims which the technique claims to support). Table 6.3 in
Chapter 6 suggests activities which are important to 5 HCI roles identified in design practice, and could be used as a framework for judging this feature.

**Presence of the Feature** was apparent for Statecharts when used as a user-oriented design specification, since the Statecharts notation did tend to support invention analysis and communication which were the main types of activity required of designers (see tables 6.3 and 6.10; Chapter 6). Statecharts did not, however support collection and analysis of the user- and task-oriented information required, and was seen to be used in tandem with TAKD by one analyst; the two techniques together providing much more coherent support for all the design activities he carried out.

**Absence of the Feature** was clearly apparent in CLG when used to support HCI consultancy on a commercial design project. It failed to support information collection and communication which were both important types of activity to the role of consultant.

Having identified ten concrete, non-overlapping features which an HCI DET may or may not possess, and which may, by their presence or absence suggest the applicability of a technique, we now need to consider a framework within which these features might be used together to assess the potential success of a candidate HCI DET as an *applied* user-oriented design or evaluation technique. In the following sections such a framework is outlined and its possible uses indicated.
8.5 AF: A Proposed "Application Framework" for Assessment of Applicable HCI DETs

In the following discussions a framework for assessing the applicability of HCI DETs is outlined in terms of each of its components; sub-sections 8.5.1 to 8.5.4, including descriptions of their respective parts and their implications for applicability. This framework presents a positive contribution to the discipline of HCI which can influence the considerations which are brought to bear on the form of future design and evaluative techniques.

The Application Framework for Assessment of the Applicability of HCI DETs (AF) contains 4 components as shown in table 8.1. Three of these components have been drawn from, elaborated and justified in chapters 1, 5 and 6, and the features list has been described in the current chapter:

The framework which is outlined in the following sections is not claimed to represent the general value of an HCI technique to the discipline as a whole. The
plausibility and psychological validity of the various claims used to support HCI DETs are not questioned (although a more detailed framework would perhaps insist that this be done). The claims made by the developers of the various HCI techniques relating to sensitivity of the technique to various properties of users and systems are taken to be true in most cases. The framework instead concentrates upon the practical issues concerning the applicability and utility of the technique for design practitioners regardless of its scientific strength.

The empirically based applied and commercial design schema (see figure 8.1) and an expanded version of the usability scoping matrix provide two different perspectives on UI design. The schema is a view of the process, and the matrix is a view of the usability issues within the scope of design. The HCI roles and activities matrix simply suggests coherent groups of activities which might be supported within a role, and the desirable features list is a checklist for an applicable HCI DET.

In the following sections the form of the framework is discussed in terms of the representations used for each of the components, and their implications. Then a number of possible roles are suggested; one of these roles presents a more stringent set of requirements for the framework than have been satisfied by this outline of it.

8.5.1 The AF Empirically Based Applied and Commercial Design Schema: Design Practice Activities, Information Sources and Design Constraints

Figure 8.1 is an adaptation of the design view presented on the basis of the two empirical user-oriented design practice studies in Chapter 5. What was observed in the two design studies will be assumed as representative of design practice in general by AF. The point of making this representation explicit is to allow the framework to be responsive to design practice, and to make more realistic statements about what type of user-oriented or HCI approach might be possible for typical design projects.

This design schema represents UI design practice in general as identified by the systems designer studies in chapters 5 and 6. The design studies suggested that the majority of projects involving UI design do not involve HCI specialists. Consequently the largest market for HCI may be non-specialists in HCI. HCI DETs could
Figure 8.1
Schema for User-Oriented Design Practice Organisation, Constraints and Information Sources Based Upon the Findings of Two Empirical Studies

<table>
<thead>
<tr>
<th>INFORMATION SOURCES OBSERVED</th>
<th>DESIGN ACTIVITIES OBSERVED</th>
<th>DESIGN CONSTRAINTS OBSERVED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifications of to-be-supported activity</td>
<td><strong>Commitment</strong></td>
<td></td>
</tr>
<tr>
<td>References and documentation on related activities</td>
<td>Explicit agreement on general goals</td>
<td></td>
</tr>
<tr>
<td>Observation of task activity</td>
<td><strong>Conceptual Specification</strong></td>
<td></td>
</tr>
<tr>
<td>Verbal task descriptions and interviews with users, experts and others</td>
<td>Defining detailed user- and functional requirements to satisfy general goals</td>
<td></td>
</tr>
<tr>
<td><em>(Possibly Parallel)</em></td>
<td><strong>Generation of Prototype</strong></td>
<td></td>
</tr>
<tr>
<td>Observation of use of prototype</td>
<td><em>Generally Evolutionary</em></td>
<td></td>
</tr>
<tr>
<td>Building functionality to satisfy user requirements, verifying UI software</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Testing</strong></td>
<td>Validating functionality with respect to requirements</td>
<td></td>
</tr>
<tr>
<td><strong>Finalisation</strong></td>
<td>Maintenance and possible further evolution</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lack of experience with interface design</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lack of experience with HCI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Complicated application or product sophistication</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lack of information about users</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Inadequate resources and undersized team</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Over-casual approach to design</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lack of assistance or collaboration from client</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lack of guidance from parties outside of team</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Over-casual approach to evaluation</td>
<td></td>
</tr>
</tbody>
</table>
be useful for supporting a wide range of user-oriented design activities including conceptual specification, prototype generation and testing which are overlapping and probably iterative. Any DET which is likely to integrate successfully into typical design practice should have application requirements which can easily be satisfied within the type of design schema represented in figure 8.1. Some examples of the measures required to do so are given below.

Design Activities

At present HCI DETs tend to focus upon conceptual specification (e.g. TAKD; Johnson et al 1984, and the top two levels of CLG; Moran 1981), or user-oriented testing design activities (e.g. TAG; Payne & Green 1986, GOMS; Card et al 1980). However support for design commitment could be a target for HCI tools which were able to suggest the types of general support a system should provide, and therefore its scope and the design goals which would have to be satisfied to provide that scope of support. As a hypothetical example; a commercial client's request for design team to produce a system which recommends books to library users might be shown to be inappropriate by a design team with the appropriate HCI DET; they might be able to show, quite convincingly that a better goal would be a support tool for the librarian to recommend books to the library borrower.

UI generation seems to involve prototyping in applied and commercial environments and could be supported by HCI DETs which capitalised on prototype generation, as well as the existence of any dialogue specifications. For example current HCI DETs tend to ignore the "look and feel" qualities of the UI which are provided by prototypes. Apart from the fact that prototyping may reduce the need for user modelling in the first place perhaps the main reasons for not exploiting prototypes, may be the difficulty which many current user models have with accounting for perceptual processing, creativity, variability, and any number of human characteristics which may become important in evaluation of simulations UI behaviour using modern prototyping tools such as Hypercard (Goodman 1987) and Trillium (Henderson 1986).

Design finalisation is also supportable. HCI DETs such as GOMS can provide system purchasers with information about various performance times for target tasks (final "beta-test" user evaluations could also provide this information). For example
Maclean et al (1989) suggest the use of a tool for providing information about design decisions and the criteria which shaped them which could help maintain user-orientedness as modifications were made to the UI by users and system modifiers. In essence this tool would represent an additional design product which could be used to ensure that the reasoning behind the design was not violated by later alterations or extensions which would endanger the consistency and coherence of the system UI.

**Information Sources**

The information sources assumed by the design schema are the ten most commonly exploited sources ranked by designers in the questionnaire study reported in Chapter 4. An HCI DET which restricted itself to requirements for these sources only would be more likely to be applicable than one which requested sources such as performance data from previous or comparable system users (as perhaps a GOMS or a CCT evaluation might). This is not to say that additional information source exploitation should not be encouraged, merely that the sources included here seem to be readily available and easy to exploit.

The information sources are represented next to the design activity within which they are usually exploited. For example, specifications of to-be supported activity tend to be available before other information. An early HCI DET would probably have to rely on this type of information, rather than the sources which are represented as being used later in the design process. However, only a relatively late HCI DET would be able to exploit information from observations of use of prototypes.

**Constraints**

Design constraints must be accommodated by applicable HCI DETs because they may render a technique completely inapplicable if they clash strongly with its application requirements (see Chapter 7). Some constraints may be more avoidable than others, but it is up to the designers rather than the HCI DET developers to judge which these are. It may be possible to avoid design constraints such as lack of experience with UI design or HCI (perhaps through self education), and over casual approaches to design and evaluation. Designers who do so may be able to use a wider range of HCI techniques. However this requires that designers make addi-
tional efforts which they may not be prepared to do. HCI technique developers cannot rely on this alone if they wish to see their methods applied in practice.

8.5.2 The AF Usability Scoping Matrix: A Representation of the Analysis Space for HCI DETs

Table 8.2 is an extended adaptation of the scoping matrix used to characterise HCI DETs in Chapter 2, and Chapter 6, and to represent user-oriented design issues in Chapter 7. The "user" evaluation factor has been expanded to enable various HCI DETs to be characterised in terms of the variety of user behaviours they can describe and explain. The greater the scope across user behaviours, the more likely it is that the DET in question can be applied to a wide variety of UI and application types (e.g. direct manipulation UIs, command line based UIs, process control applications, and CAD applications). At present the concentration on text editing systems, "email" systems and their kin by HCI DETs enables them to avoid coming to terms with the implications of tasks which involve wider demands on human abilities (see Chapter 7, Section 7.2).

A generalizable scoping matrix must include these user sub-factors which may be inadequately dealt with by existing HCI DETs. However, as stated earlier in this chapter, these sub-factors were not applied in the characterisations of the HCI DETs reviewed in this thesis since their attention has been largely limited to knowledge representation, language and proceduralised behaviour with limited considerations of errors in response to certain properties of systems.

The other factors might also be expanded where appropriate, for example the application factor might be expanded into sub-factors such as state-behaviour (such as modality, legal operations and broadcasting of effects of operations), architecture, hardware components, and so on. However, the focus of the approaches reviewed in Chapters 2 and 6 has been largely upon user-orientedness in design and the complexities of other aspects of system design have been avoided since they are less directly relevant to HCI.

The scoping matrix does not directly suggest the nature of the users, UI, application and target tasks which an HCI DET could be appropriate for. In order to ascertain this, an analyst would have to assess the nature of the system he or she was design-
Table 8.2
Expanded Scoping Matrix of Classes of Usability Principle Over Evaluation Factors

<table>
<thead>
<tr>
<th>Evaluation Factors &amp; Subfactors</th>
<th>Usability Principles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simplicity</td>
</tr>
<tr>
<td>Users (Us)</td>
<td>Learning</td>
</tr>
<tr>
<td></td>
<td>Perceptual Processing</td>
</tr>
<tr>
<td></td>
<td>Knowledge Rep'n</td>
</tr>
<tr>
<td></td>
<td>Language</td>
</tr>
<tr>
<td></td>
<td>Planning</td>
</tr>
<tr>
<td></td>
<td>Problem Solving</td>
</tr>
<tr>
<td></td>
<td>Proceduralised Beh'Y</td>
</tr>
<tr>
<td></td>
<td>Errors</td>
</tr>
<tr>
<td></td>
<td>Social Behaviour</td>
</tr>
<tr>
<td></td>
<td>Organisational Beh'Y</td>
</tr>
<tr>
<td>System Application (App)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>User Interface (UI)</td>
</tr>
<tr>
<td></td>
<td>Target Tasks (TTs)</td>
</tr>
<tr>
<td></td>
<td>Acceptability of Performance (Acc)</td>
</tr>
</tbody>
</table>
ing in terms of these factors, and determine which cells were the most important to the usability of the system. He or she would then need to assess whether a technique was able to address these issues, and determine whether there was a good match between what the design project required, and what the HCI DET provided. Note that it is possible to expand the contents of each cell to give either a weighting of its importance to a design project, and another weighting to the degree of rigour or accuracy with which it is represented, or a description of how it is tackled by a technique.

The analyst using the framework to characterise an HCI DET would have to judge whether a technique meets the requirements for capturing each usability principle (a to f below) with respect to the evaluation factors (g to k below). A Brief reminder of the implications of row and column headings for table 8.2 (based upon table 2.1, Chapter 2) follows.

Models which capture:

(a) Simplicity must model (and present metrics for) simplicity of representations or specifications.

(b) Compatibility must capture the relationships between external (real world, and other systems) knowledge and the knowledge required to use the system.

(c) UCTDs must present system or device independent task analyses and goal structures which reflect the needs and characteristics of the user.

(d) (e) & (f) Observability, Consistency and Retrievability must have some component for presentation of user independent representations of the appropriate system properties.

Models which account for:

(g) Users must employ empirically based psychological theory or preferably explicit models of human information processing.

(h) System Applications must entail representations of system functionality and
behaviour.

(i) **UI** must be sensitive to static and dynamic representation and behaviour in interaction (e.g. command versus menu driven).

(j) **Target Tasks** must specify a coherent (or perhaps representative) set of basic or composite tasks the system is designed to support.

(k) **Acceptability of Performance** must generate testable predictions about qualitative or quantitative aspects of user behaviour.

The way in which the principles of usability might be addressed in terms of each of the evaluation factors was elaborated in more detail in Chapter 2. An analyst interested in applying this matrix to a design project and an HCI DET, to determine whether the latter was appropriate for the needs of the project, (see the example in Chapter 7) could use elaborations for each cell based upon the explanations and justifications of the characterisation for each technique reviewed in Chapters 2 and 6.

However table 8.2 contains the additional user evaluation sub-factors which make the assessment of any technique potentially far more complicated. The human sub-factors may be captured by a technique implicitly or explicitly, and the ability of an analyst to achieve this sensitivity may depend more on the skills of the analyst than the technique itself. For example Reisner (1981) points out the possibility of placing semantic restrictions on some of the rules in the Action Language for the "Robart" drawing tools. These restrictions would help to avoid redundancy in the rules by allowing the grammar to show generalisations across similar rules as users might represent them. The semantic restrictions produce a more psychologically valid rule set and increase the user knowledge representation powers of the notation. However it would probably take a considerable amount of HCI skill to carry out such extensions to Reisner’s approach.

Given the potential complexity of the scoping matrix, the human sub-factors are not judged with respect to each of the usability principles. This is because many of the principles are not even relevant to some of them, and also because it may be difficult to prove whether a model really does embody each principle with respect to
each particular sub-factor.

By adding the social and organisational sub-factors, a number of additional usability principles might be suggested which would be important to these views of a UI (two examples of additional principles might be "social responsiveness"; the system behaves more like a human personality and "business efficiency"; the system actively prevents the user wasting company time). Likewise additional evaluation factors might be added to the existing ones, such as "production costs" and "market". However these are not clearly within the domain of HCI, and are therefore not presently included. The absence of such principles, however, is not a serious problem for the matrix since it can clearly be expanded where appropriate to address new principles and sub-factors.

It was noted in Chapter 6 that the technology transfer technique was probably not adequately captured by the existing Scoping Matrix because it aimed to encourage designers to consider business and marketing aspects of systems which could require expansions in terms of additional principles and evaluation factors. This indicates, either an inadequacy of the Matrix in its present form (which would have to be expanded), or a mixing of concerns in the technology transfer technique, with business interests being combined with HCI interests.

The scoping matrix is not by any means a formal tool and may be viewed as a memory aid which encourages the analyst to keep a broad view of potentially interesting or important perspectives from which a design might be viewed. The matrix may even be expanded to include principles peculiar to a particular project, such as compatibility with a particular in-house style, or additional evaluation factors which have not been included in the matrix, such as the cost of the design and evaluation, or the state of competition in the market for the product of the design.

8.5.3 The AF HCI Roles and Activities Matrix: A Representation of Coherent Support Options

Chapter 6 suggested a number of HCI roles which act as an organising scheme for types of activity. Table 8.3 is derived from table 6.3; it represents the need for support for particular types of activity associated with HCI specialists roles. These roles would also be relevant to systems analysts and engineers charged with design
Table 8.3
Matrix of Support Requirements for Activities Involved in
the Various Roles of HCI Specialists in the Study

<table>
<thead>
<tr>
<th>Roles</th>
<th>Activity types</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Information Collection</td>
</tr>
<tr>
<td>Self Educator</td>
<td>S</td>
</tr>
<tr>
<td>Designer</td>
<td>s</td>
</tr>
<tr>
<td>Researcher</td>
<td>S</td>
</tr>
<tr>
<td>Consultant</td>
<td>S</td>
</tr>
<tr>
<td>Technology Transfer Agent</td>
<td>s</td>
</tr>
</tbody>
</table>

S = Requires Strong Support
s = Requires Little Support

of, and evaluation of the usability of, a UI (Chapter 5 showed systems designers tended to be responsible for all aspects of the system including the UI).

The roles identified involved a variety of types of activity (see table 6.3). These activities are distinguished from the design goal-oriented activity classification in the design schema (see Chapter 5) by the fact that they are classified according to more basic or general aims (such as information collection, and analysis). So, for example, the conceptual specification activity type in the design schema could involve many or all of the activity types involved in the various roles. The precise UI and HCI oriented activities suggested by the activity type in table 8.3 are roughly
classified as; Information Collection (IC), Invention (I), Analysis (A), and Communication (C), or some combination of these.

Reference to literature and reports (IC)
Attendance of courses, workshops, conferences, meetings etc. (IC)
Interview (IC)
Observation (IC)
Empirical evaluation (IC) & (A)
Task analysis (A)
User requirements analysis (A)
Analysis of requirements and functional specifications (A)
Design/evaluation technique application (A)
UI design (I)
Design/evaluation technique development (I)
Creative solution generation (I)
Functional specification of UI (C)
Demonstration (of techniques or products) (C)
Running workshops, giving seminars or presenting papers (C)
Writing reports/documentation (C)
Correspondence (verbal, electronic or on paper) (C)

The roles in which these precise activities play an important part seem for the most part to be intuitively obvious, but are described throughout Chapter 6. However, it is left to the analyst using AF to judge the precise activities which subsume the roles which are important in the project(s) for which an HCI DET is being designed or chosen, and to determine whether the HCI DET supports all of these. The activity support matrix is really an outline guide to the sorts of activity which are likely to be most important to each role.

It is clearly important that an HCI DET supports as many activities as possible, if it is to serve as a multi purpose design tool. For example Moran (1981) suggests a number of potential activities CLG could be used to support (it could be used both for design and evaluation, for competence, and even for performance modelling with some extensions). However, to avoid being unnecessarily cumbersome it may be necessary to restrict the support a technique provides to a small set of activities. It would seem sensible to aim for a set of activities which supported an HCI special-
ist or a UI designer throughout a project, thus removing the need to learn to use more than one technique. So a technique which restricted itself to activities involved in one role would tend to be more applicable than one which supported only one type of activity, since perhaps three or four of the latter type would have to be learnt by one individual with a particular role.

8.5.4 The AF Desirable Features List

Table 8.4 shows the desirable features list, the items of which were defined for the purpose of detecting their presence or absence in an HCI DET in section 8.4. How important they are in practice may be influenced by the particular circumstances in which a technique is used; for example lack of simplicity may not be a problem for design projects with generous resources and expertise in HCI. Narrow scope may not be important if the issues which are addressed by a technique are overriding important, and no other technique deals with them so well.

The desirable features (see table 8.4) are deliberately presented as the last part of an AF analysis, since the preceding three sections (the design schema, the scoping matrix, and the support matrix) are used to formulate any information in such a way that it is easier to determine the existence of some of these features in an HCI DET.

By evaluating an HCI DET in terms of these features it may be possible to highlight aspects which will prevent it from being applied in particular circumstances where particular features are important. Looking at an individual project, it should be possible to determine whether some features are more or less important, given available resources, expertise, information sources, and so on.

8.5.5 Possible Roles of the Application Framework: What the Framework is, and What it Can Do

AF has 4 components which each provide either a view, or a structure for a view, of some important aspect of user-oriented design practice; important that is for HCI DETs.

The Design Schema presents a realistic picture of the type of situation an applicable HCI DET should be designed to deal with.
Table 8.4
Desirable Features of Applicable HCI DETs

Simplicity
Scope
Explicit Entities and Procedures
Expressiveness
Visual Complexity
Broad Behavioural Scope
High Generality
Realism of Design View
Preservation of Important Information
Coherence of Support

*The Scoping Matrix* presents a space within which any particular design project or HCI DET can be characterised. In the former case it would represent areas that required special analytical attention (see the example in Chapter 7), and in the latter it would represent areas which a particular technique is capable of addressing (as in chapters 2 and 7).

*The HCI Roles and Activities Matrix* suggests coherent support for HCI roles which an HCI DET might attempt to follow.

*The Desirable Features List* makes explicit a set of potentially evaluable features which are desirable for an applicable HCI DET. These features are based upon the findings from the three design studies reported in this thesis.

Having discussed the form of the framework, the uses to which it could be put will be addressed. These possible roles would require few significant modifications to the framework which is essentially intended to act as a view of design with respect to HCI techniques. Four possible uses are suggested in the following.
An HCI Design View
The AF provides a view of design practice as it relates to the discipline of HCI. In other words it is a deliberately distorted picture which emphasizes certain aspects which usually receive scant attention (such as the activities and roles of HCI specialists, and the design constraints which interfere with usability). This is perhaps the most straightforward role for the matrix and the one which has most nearly been fulfilled in the preceding subsections.

An HCI DET Development Guide
On the basis of the research presented in this thesis, AF emphasizes those aspects of design which appear to need to be addressed by applicable HCI DETs. Researchers in the field of HCI wishing to develop tools and techniques for use in applied and commercial design practice may find the framework useful as a guide to development of an applicable technique, as opposed to a technique whose main value is towards advancing theory of HCI.

AF will not provide support for the development of basic concepts and components (such as a cognitive architecture, a system simulation or model, or a notation with the appropriate formal power or psychological validity) it should, however suggest the questions which must be addressed if the (theoretically validated) technique in question is ever to prove usable to analysts other than the developers of the technique itself, and able to work in an applied system design project.

A Framework for Critiques of HCI DETs
Just as AF helps to raise questions for the developer of an HCI DET, it should suggest ways in which techniques can be criticised by others. It provides a useful basis for comparison of two or more techniques in terms of their usability and applicability.

An HCI DET Selection Device for applied and commercial designers
For systems designers AF could, work as a highly generalizable, simple framework for selecting an appropriate HCI DET for a design project. The system designer attempting to use it for this purpose would have to characterise both the design project in question, and the HCI DETs of potential interest, in terms of the framework. Of course the design schema, in this case would have to be transformed to match any projected approach or expected constraints and information sources. Likewise
the roles and activities matrix would have to be modified to fit expected roles and activities for the project in question.

By following the same schematic forms the designer could produce a simple design outline which could be balanced against the DETs application requirements (see Chapter 7), and a matrix reflecting typical roles assigned within the host organisation of the project and suggesting the amount of support required for each type of activity within a particular role (i.e. the importance of the activity type to the role in question).

The default design schema presented in figure 8.1 has been developed on the basis of evidence from a highly variable set of system design projects described in three studies (see chapters 4, 5 and 6) involving systems designers and HCI specialists. For this reason it is firmly rooted in practice, as opposed to theory and should be easily adapted to many projects. The same could be said of the roles and activities support matrix. It should also be easy to adapt this to suit a given project if the roles and activity types are generally consistent within the organisation where the design project is to take place.

The main difficulty with this selection role is that it requires that the designer or analyst applying the framework has sufficient knowledge of all the HCI DETs he or she is considering as candidates in order to characterise them in terms of the usability scoping matrix. As has been stated repeatedly in this thesis, many systems designers have little or no familiarity with HCI techniques. It would therefore be infeasible to expect them to familiarise themselves with a number of potentially inapplicable techniques in order to choose one which actually is applicable to their project.

Clearly HCI techniques must provide explicit information of this type in order to allow potential technique appliers to make an informed choice as to which technique most suits their requirements. This is not an unreasonable request if HCI wishes to be taken seriously by designers in applied and commercial projects who do not have the time to spare to read many papers relating to a discipline in which they are not specialists.
8.6 Illustration of Application Framework's Descriptive Power

Perhaps the best way of elucidating AF is to provide a worked example of how it might be applied to a real design project. The role focused upon in the example is that of an HCI technique selection device. This is the most demanding role as it requires that the framework be adapted to represent more accurately the particular design approach to be used (if this is known) and the probable roles and activities involved in the project (if these are also known about). All other potential roles of the framework are envisioned as requiring little or no adaptation, unless the scoping matrix needs to be expanded for a particular focus of some DET.

It is important to note that project 5 is of course a completed design, and as such is not an ideal target for real application of AF. For this reason, a starting assumption in this example will be that the host organisation always takes the same general approach to design projects and designers and other individuals adopt predictable roles. In this sense the general course and roles of the project are reasonably predictable (bearing in mind unpredictable low level implementation difficulties, bugs etc).

8.6.1 An Example of Application Framework: 
CCT and the Workstation Window Manager

In Chapter 7 a worked example was presented in which the scope of CCT was matched against the scope of the important issues for one of the design projects described. Here, the same example, referred to from here on as EX (for example), is expanded into a complete AF analysis which would suggest the feasibility of applying CCT in project 5 from the Design Interview study reported in Chapter 5. As a reminder, project 5 involved the following:

A window manager to be used by programmers, produced by a team of a systems architect, a software designer, and a consultant with HCI experience. The window manager was designed for a Unix workstation with a graphics display. The HCI consultant provided a catalogue of interactive techniques recommended for good interfaces. The other two team members did not have a great deal of HCI experience themselves.
Figure 8.2
EX Schema for Design Practice Organisation, Constraints and Information
Sources Based Upon Project 5 from the Designer Interview Study

<table>
<thead>
<tr>
<th>INFORMATION SOURCES EXPECTED</th>
<th>DESIGN ACTIVITIES EXPECTED</th>
<th>DESIGN CONSTRAINTS EXPECTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commitment</td>
<td></td>
<td>Lack of experience with HCI</td>
</tr>
<tr>
<td>Existing comparable systems</td>
<td>Explicit agreement on general goals and functional requirements</td>
<td></td>
</tr>
<tr>
<td>Specifications of to-be-supported activity</td>
<td>Defining detailed interactive methods, and functional requirements to satisfy general goals</td>
<td>Lack of information about users</td>
</tr>
<tr>
<td>Documentation on related activities</td>
<td>Building functionality to satisfy modifiability requirements, interaction methods, and verifying UI software</td>
<td>Over-casual approach to design</td>
</tr>
<tr>
<td>(Parallel)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Observation of use of prototype</td>
<td>Validating functionality with respect to requirements</td>
<td>Over-casual approach to evaluation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finalisation</td>
<td>System marketed and maintained with the workstation on which it runs</td>
<td></td>
</tr>
</tbody>
</table>
I. Adapting the AF View of Design Practice:
Design Practice Activities, Information Sources and Constraints

AF is to be used as a selection device for a particular project EX. This means that the design schema needs to be adapted to fit more closely with the particular characteristics of typical projects in the host organisation for the project. The starting assumption is therefore that it is possible to predict certain characteristic design activities (or lack of them), information sources, and constraints for EX. This is done on the hypothetical basis of assumed 'normal practice'. In this example the assumptions about the design process and the roles and activities involved have to be taken from what transpired in the original project 5 itself, since we have no other information about projects in the host organisation.

If in reality it was impossible to predict the course of a design project (as perhaps it might be if a completely novel application was being generated) AF would recommend that the design schema and the roles and activities support matrix default to the empirically based general versions (in the case of the roles and activities support matrix this might still have to reflect the nature of the design team) as presented in figure 8.1 and table 8.3.

We shall assume that the design team in EX are considering CCT as a possible candidate for supporting the design and evaluation of the UI in their project. We shall also have to assume that they have information relating to the design expectations and requirements of CCT (which were outlined in Chapter 3).

Figure 8.2 represents the expected EX design process.

The expected information sources are:
* Existing comparable systems; (these influenced all of the team members throughout project 5. The systems architect conceived the system on the basis of comparable window management systems on similar workstations to the one for which this software was being designed).
* Specifications of to-be-supported activity; (provided by the EX team itself).
* Documentation on related activity; (in project 5 this was provided by the HCI consultant after initial system goals were established. It was a document containing descriptions of interactive methods for graphics workstations).
* Observation of use of prototype; (in which the EX designers use themselves as subjects. No representative or outside users are tested the prototype).

The expected design activities are:

* Commitment; (in project 5 one of the designers (the systems architect) decided to outline a system, and then communicated the system structure to the software designer and the consultant).

* Conceptual Specification; (particular interactive methods were selected from the HCI consultant's document in project 5).

* Generation; (in which the EX software is implemented in modifiable low level components to support the interaction and allow iteration and changes to be made).

* Testing; (is carried out exclusively by the EX software designers who pretend to be naive users).

* Finalisation; (involves the sale of the system together with the EX workstation on which it runs).

The expected design constraints are

* Lack of Experience with HCI; (in project 5 only the HCI consultant had such experience).

* Lack of Information about Users; (no attempt is made in EX to carry out empirical studies or read about user requirements).

* Over-casual Approach to Design; (the host organisation of EX does not encourage a structured approach to design).

* Over-casual Approach to Evaluation; (the host organisation does not encourage a structured approach to evaluation).

CCT requires mainly the following information sources:

* Empirical user performance data NOT PROVIDED IN EX

* User cognitive simulation NOT PROVIDED IN EX

* GTN simulation of UI behaviour NOT PROVIDED IN EX

* Specification of Target Tasks INFORMALLY PROVIDED IN EX
CCT assumes the following design phases
(all of which may be iterated; Polson 1987; see Chapter 3).

* 1. System Definition Phase: Specify the functional requirements of the system.
   EQUIVALENT TO CONCEPTUAL SPECIFICATION IN EX

* 2. Task Analysis Phase: Specify the user’s decomposition of each task performed by the system.
   NOT EQUIVALENT TO ANY PHASE IN EX

* 3. Detailed Design Phase: Specify the details of the user interface, including methods, menus, commands, etc.
   POSSIBLY EQUIVALENT TO GENERATION IN EX

* 4. Evaluation Phase: Evaluate the design using simulation methods and modify on the basis of evaluation.
   NOT EQUIVALENT TO ANY PHASE IN EX

The CCT Approach does not assume any design constraints, however there are 4 here which strongly impinge upon the probable success of CCT.

* Lack of Experience with HCI; means that the EX systems designers will probably have great difficulty understanding CCT

* Lack of Information about Users; means that without taking a significant amount of extra time to set up user studies CCT cannot be applied in EX.

* Over-casual Approach to Design; means that no coherent specification of target tasks for the system is provided in EX.

* Over-casual Approach to Evaluation; means that the EX designers are not overly concerned about the usability of the system, assuming users will, as frequent users, quickly find it as simple as they do.

The CCT view of the design process is not very close to the reality in the EX example. It expects many extra information sources to be exploited which would clearly be expensive for a small design team (as project 5 had). It has a more complicated model of design phases which does not include initial and final phases (commitment and finalisation) which could also require HCI support. It clashes quite markedly with each of the constraints in EX.
This suggests that in order to apply CCT to EX it would be necessary to make significant changes to the host organisation design approach. It seems less likely that this would happen than that a simpler, perhaps less powerful HCI approach would be chosen in preference to CCT, on the basis of the incompatible approaches they have to design.

II. The AF Scoping Matrix:
A Representation of the Analysis Space for HCI DETs

In chapter 7 a similar scoping exercise for CCT and project 5 was carried out, which showed the similarity between the issues important to the window manager, and those within the predictive scope of CCT. For a more detailed explanation of the features which place CCT and EX in the cell in which they are shown, see Chapter 7. Suffice it to say here that, on the basis of its characterisation according to the usability scoping matrix, CCT represents the closest match, amongst the HCI DETs discussed in this thesis, in terms of scope, to the requirements of project EX (for comparisons see other techniques’ scoping matrices in Chapters 2 and 6).

The only new addition to the scoping matrix shown in table 8.5 is the set of user sub-factors which reflect the requirement stipulated in the desirable features list that the scope of user behavioural characteristics be as wide as possible to allow generalisation to a variety of UIs, applications, and target tasks.

The system in question in EX is a direct manipulation graphics workstation window manager, to be used by relatively skilled system users. Although CCT deals well with skilled (proceduralised behaviour) which is obviously important to EX, it has little power to capture perceptual behaviour, which would also be highly important for EX which requires the user to interpret patterns, movements, changes in size of objects on the screen, and so on. CCT contains no perceptual processing components within its cognitive architecture which would enable it to distinguish one set of icons from another, or one method of indicating a window was being moved from another method.

In respect of the expanded user sub-factors part of the scoping matrix, the apparent match between CCT and project EX with respect to users suggested in Chapter 7, is not quite as close as it might at first seem. CCT would probably provide no assis-
<table>
<thead>
<tr>
<th>Evaluation Factors &amp; Subfactors</th>
<th>Usability Principles</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simplicity</td>
</tr>
<tr>
<td>Learning</td>
<td>CCT &amp; EX</td>
</tr>
<tr>
<td>Perceptual Processing</td>
<td>EX</td>
</tr>
<tr>
<td>Knowledge Rep'n</td>
<td>CCT &amp; EX</td>
</tr>
<tr>
<td>Language</td>
<td></td>
</tr>
<tr>
<td>Planning</td>
<td></td>
</tr>
<tr>
<td>Problem Solving</td>
<td></td>
</tr>
<tr>
<td>Proceduralised Behv</td>
<td>CCT &amp; EX</td>
</tr>
<tr>
<td>Errors</td>
<td>EX</td>
</tr>
<tr>
<td>Social Behaviour</td>
<td></td>
</tr>
<tr>
<td>Organisational Behv</td>
<td></td>
</tr>
<tr>
<td>System Application (App)</td>
<td>EX</td>
</tr>
<tr>
<td>User Interface (UI)</td>
<td>CCT</td>
</tr>
<tr>
<td>Target Tasks (Tts)</td>
<td>CCT</td>
</tr>
<tr>
<td>Acceptability of Performance (Acc)</td>
<td>CCT</td>
</tr>
</tbody>
</table>
tance with the design of the system presentation. As it turned out, the main problem
with the project 5 design was overloading of the window manipulation icon (dif-
ferent combinations of mouse clicks on this icon produced different responses).
This problem seems to be to do with the confusability of the mouse actions, com-
bined with the automated nature of user's actions with the mouse (it was simply
very easy to make a slip-up). Since CCT does not address errors in great detail it is
doubtful whether it would have picked this up. On the other hand, errors were not
perceived as a major concern with the project because no serious consequences
were likely to emerge from them (for example one might find oneself shrinking a
window accidentally instead of moving it). The designer interviewed, only realised
the importance of this annoying problem after release of the product.

III. The AF Roles and Activities Matrix:
A Representation of Coherent Support Options

Since EX is a project where roles, like design processes, are assumed to be predict-
able, the default roles and activities support matrix shown in table 8.3 has been
adapted to fit assumed roles and activities in this instance.

The designers (and consultant) in EX are assumed to be responsible for all aspects
of the design and evaluation of the system. They therefore need to undertake all
types of activity in the support matrix. The support matrix implies the basic activi-
ties which are presented in table 8.6 roughly classified as Information Collection,
Invention, Analysis, and Communication, or some combination of these. The ones
carried out in EX are in bold print.

The designers in project 5 carried out a small amount of information collection from
meetings, and a good deal of design. The HCI consultant carried out some analysis
in preparation of the interactive methods document, and some communication in
presentation of the document in paper form. The systems designers tended to re-
strict their activities to invention (the only information they collected was from the
consultant and in design project meetings). The consultant carried out information
collection, analysis and communication.

Assuming the same activities in EX as in project 5, the support matrix in table 8.7
suggests that CCT does not strongly support either the activities in the designers'
Table 8.6
List of HCI-Oriented Applied and Commercial Design Practice Activities from Chapter 6

INFORMATION COLLECTION
Reference to literature and reports
Attendance of courses, workshops, conferences, meetings etc.
Interview
Observation
Empirical evaluation (for data)

ANALYSIS
Empirical evaluation (for explanation)
Task analysis
User requirements analysis
Analysis of requirements and functional specifications
Design/evaluation technique application

INVENTION
UI design
Design/evaluation technique development
Creative solution generation

COMMUNICATION
Functional specification of UI
Demonstration (of techniques or products)
Running workshops, giving seminars or presenting papers
Writing reports/documentation
Correspondence (verbal, electronic or on paper)

roles or the consultant’s role. CCT would not support information collection as it is not clear about the details of selecting appropriate subjects and experimental circumstances for deriving operator times, target task structures and so on. It is reactive to design and may only provide limited support for invention; Polson’s (1987) claim that the technique can be used for early evaluation depends upon the design process achieving a workable simulation of all the necessary functionality in the early stages. This seems to be an unlikely possibility for EX where resources are not suitable for simulations to be built.
For analysis, the technique enables predictions about learning and performance to be generated (although whether these are important metrics in EX is perhaps another question). CCT does not explicitly support communication of its user model and the implications of the mismappings between system and user goal hierarchies. However Kieras and Polson (1985) make the unsupported claim that the GTN device models can be used for communication of the system behaviour to users to help them build an appropriate model of it.

Table 8.7
Matrix of Support Requirements for Activities Involved in
the Various Roles of the Designers in EX
Including Assessment of CCT Support

<table>
<thead>
<tr>
<th>Activity types</th>
<th>Information Collection</th>
<th>Invention</th>
<th>Analysis</th>
<th>Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roles</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Designer</td>
<td>S</td>
<td>.</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>Consultant</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>CCT Support</td>
<td>No</td>
<td>No ?</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

S = Requires Strong Support
s = Requires Little Support

IV. The AF Desirable Features List

Project EX has few of the extra resources and skills available to it to enable it to ignore some of the setbacks which could be imposed by a lack of AF "desirable features" (see table 8.4). For the purposes of argument we shall assume that all of
the desirable features apply to CCT in this instance, and will briefly discuss each one in turn. However, in certain circumstances some features would be less important that others and features could be traded off against one another, given that, in an imperfect world, it will be difficult to satisfy all desirable features at once.

It should be noted here that the role of AF is not to direct the development of improvements to CCT (and all HCI DETs); this task would be immense. AF restricts itself to highlighting areas where CCT is likely to be weak with respect to applicability. It must remain the job of the developers of techniques to solve the particular application problems of their own approach.

**Simplicity:** it seems clear that CCT is not simple to apply. It requires three models; of the user, the UI and the target tasks, to be implemented and combined in a simulation. Prior to this empirical data has to be collected to support the task model (unit task times). After running the simulation, it may be necessary to redesign the simulation and then run it again.

**Scope:** CCT does not address the underlying application as part of its scope (see Chapter 2). It's device model is a superficial, and selective representation of UI behaviour. It so happens that this is appropriate for EX because the application is a window manager which manifests most of its behaviour as changes in presentation anyway.

**Explicit Entities and Procedures:** CCT is particularly inexplicit as to how production rules should be organised to produce simulations of interactive behaviour. The designer writing the simulation could produce a completely arbitrary production system which had little to do with the spirit of ACT* (Anderson 1983; see Chapter 2).

**Expressiveness:** the expressiveness of CCT is entirely dependent upon the representativeness of whatever criterion level of real user performance it accepts after some expected training. If this selected criterion level is realistic, and representative of the performance it predicts then CCT may produce accurate speed, or subsequent learning predictions. Without explicit knowledge of what the system users know when they begin to use the system (and how they represent it) CCT is likely to become highly inaccurate. If no training is guaranteed for users,
then this is likely to be case. The sensitivity of metrics to UI and task variations, and to practice obtained by its authors could only be guaranteed in situations where rigid training programmes for system users were enforced. In EX it is assumed that no training is planned for the system users.

**Visual Complexity:** Although not as complicated as CLG in terms of notational devices and use of bracketing, the GOMS style notation of the ACT* production rules, and the goal hierarchy notation, the GTN notation together, make CCT specifications rather diffuse and impractical to view in a holistic manner. On the other hand, use of indentation in the GOMS specifications makes task, and sub-task identification relatively easy. Task-to-Device mappings are based upon simplifications of the system state representations and the user’s task goals. Since much redundant information is removed, these seem to be a useful simple summary of system-user mismatches which many other DETs fail to provide.

**Broad Behavioural Scope:** the behavioural scope of CCT is perhaps surprisingly narrow (a problem which many HCI DETs may share) it fails to capture perceptual behaviour (table 8.5 suggests that CCT cannot deal with observability with respect to users) and error behaviour, both of which are relevant to EX (although errors were not of great importance in project 5 because of their limited consequences).

**High Generality:** due to its limited behavioural scope, it seems likely that CCT (like many other HCI DETs) is not likely to be highly generalizable to a wide variety of system types. Any claims regarding this matter would, however, have to be empirically vindicated since it may be that the behavioural scope it does have is sufficient to allow its generalisation to a number of applications (even if it cannot in practice be applied). Given the limited available information here it is difficult to tell if CCT is really appropriate for analysis of the type of UI involved in EX. It lacks the ability to capture the important human perceptual characteristics of behaviour, but it could be sensitive to other aspects of interactions such as ordering in task sequences, or mousing differences in menu versus icon-based window control.

**Realism of Design View:** In this instance the CCT design view fails to map to the realities of EX. The design process aspects of EX are perhaps the most problematic area for CCT (see above) since most of its assumptions fail to be satisfied.
Preservation of Important Information: The obvious problem in EX is that CCT will not indicate any likely error sites which are dependent on similarities of action sequences between similar tasks, rather than poor task-to-device mappings. Hence it seems unlikely that it would spot the problem with the overloaded window icon. In other words the simulation could produce a false sense of security for the designers in EX by only highlighting some potential user difficulties, whilst obscuring others.

Coherence of Support: As shown above CCT does not seem to have a coherent approach to support for the person using the technique. The roles identified in EX involve different activities, and (although it is probable that many useful activities are not carried out by the design team) neither of the roles is supported by CCT.

The above is by no means a complete analysis of CCT's embodiment of desirable features for application. It is brief, and selective with the emphasis on major failures of the technique. It merely serves to suggest how AF may reveal some important reasons why CCT could prove very difficult to apply within a design project like EX.

Overall, then it would seem that the impression gained from the use of the scoping matrix alone (see Chapter 7) was rather misleading. CCT appeared to be a very suitable match for the requirements of project 5. It turns out in the EX example however, that CCT has many practical problems associated with it and AF indicates that it would be highly unwise to select it for such a project. This demonstrates the importance of using the framework as an integrated whole since applying only one component could give a very misleading picture of the applicability of a particular technique.

8.6.2 Summary of the Application Framework for Assessing HCI DETs

The brief example above demonstrates the AF role of HCI DET selection device for a systems design project. This role is probably the most complex and unrealisable for the framework because it required adaption of the design schema and the roles and activities support matrix to suit the particular conditions of the design project concerned.
The other uses of the technique would normally exploit the original design schema and support matrix. Their application of the framework would be relatively straightforward, and would have a great deal in common with the example illustrated above. An outline of each of the other three roles follows.

**An HCI Design View**

The AF provides a deliberately distorted picture of design practice, as it occurs in applied and commercial environments, which emphasizes certain aspects which usually receive scant attention (such as the activities and roles of HCI specialists, and the design constraints which interfere with usability). As a design view it may be used by HCI researchers as a brief picture of the practical implications of design practice for HCI intervention, either as practised by consultants, or as outlined or recommended in user-oriented methodologies.

**An HCI DET Development Guide**

Although the AF will not provide support for the development of basic concepts and components it should, suggest the questions which must be addressed if a technique is to prove usable to analysts other than its developers. It suggests directions which the developers of HCI tools and techniques might most profitably take in improving the impact which they have on design. The desirable features list could be used as a set of principles which should be adhered to in the development of the new HCI DET. By ensuring that application requirements which a DET possesses are likely to be satisfied by design practice with its typical constraints, and by attending to the desirable features list, an HCI DET may be better designed for practical use by systems designers or HCI practitioners in commercial environments.

The only modification envisioned for the AF within this role is the adaptation of the usability scoping matrix, to address principles or evaluation factors which are not currently included as suggested earlier in this chapter. The scoping matrix would help to remind the developer of the breadth of the technique which would be a useful perspective to consider, and contrast with the expected effort required to apply the technique. Narrower scope also suggests that a technique is likely to satisfy the analytical requirements of fewer design projects.

**A Framework for Critiques of HCI DETs**

The role of AF as a framework for critiques of HCI techniques is dependent upon
the assumption that the person carrying out such a critique is a specialist in HCI, or is familiar with the technique(s) being assessed. If this is not the case it could be difficult to carry out a representative scoping exercise as many HCI DETs are not explicit about their scope and do not advertise their limitations. Since the framework's development has been driven by studies aimed at analysing why HCI DETs are not applied, and at what features are desirable for applicable techniques, it is strongly oriented towards this role.

In the example presented in the previous section, the assessment of CCT in terms of the AF was tailored towards a particular project. However, a very similar analysis could have been carried out to obtain a general application critique of the CCT approach, by substituting the default design schema and roles and activities support matrix for the adapted versions of these components. Comparisons between several approaches with equivalent scope but varying applicability would also be possible.

In order to be consistent with its own philosophy, it could be argued that the Application Framework presented above should exhibit the features, or their equivalent, which it states that an applicable technique should embody. If it does not do so then it has shortcomings as an applicable technique for applied and commercial practitioners itself. However this is only one of the possible roles of AF; the main role perhaps being that of a design view for HCI.

On the other hand, a useful structure for the discussion of this framework will be to address the existence or lack of AF's desirable features for application to AF itself. The following summary restates the main features of the framework and represents a brief attempt to address how well its components conform with the ten desirable features discussed in section 8.4.

Simplicity of the AF framework relates to what the Analyst is expected to do in order to apply it. The framework may be seen as a design view, an HCI DET development guide, a basis for critiques of DETs, or a selection device for applicable techniques in design projects. In Chapters 7 and 8 the design schema, the scoping matrix, the support matrix and the features list have all been demonstrated. The framework is relatively straightforward and can be adapted to suit the skills and needs of its user. The most complex aspect is perhaps the scoping matrix which requires a working knowledge of the character of an HCI DET. This is not a failure
of the framework itself, rather it is related to the lack of explicitness of most HCI DETs in stating clearly and concisely what their scope in design might be. For example Moran (1981), Card et al (1983), and Kieras and Polson (1985) attempt to suggest some practical aspects of the scope of their approaches, but these are presented in a diffuse and confusing manner which requires a potential applier to read, perhaps several papers, carefully before a comparison can be made between the techniques.

Scope is an important issue for AF. It attempts to capture a wide scope of possible system properties from a number of viewpoints. Whether the scoping matrix requires further expansion or not probably depends upon the views of any who might see it as useful. For example, as a selection device for an SADM, AF is clearly too limited to HCI concerns.

Explicit Entities and Procedures need to be presented in order that an analyst can take up a method and use it appropriately. AF is not presented here in a clear enough manner to ensure that it could be used by anyone, however it is hoped that the demonstration of its application would be sufficient guide for most.

Expressiveness of AF is perhaps not an important feature as it does not claim to capture any precise theoretical property of design practice. Its roots are based in empirical observation, but the features of design which it embodies are open to question and the approach may be adapted or expanded to suit the particular needs of anyone using the framework.

Visual Complexity of the framework is kept to a minimum in AF. All of its devices are based upon tables or lists (of activities or desirable features). Of course the framework has no formal properties which the notation might otherwise need to preserve, and therefore its simplicity is largely due to the lack of a strong requirement for notational precision.

Broad Behavioural Scope for an HCI DET refers to the wide range of properties of the thing that might be modelled (i.e. the user and his/her interaction with a system UI). The four components of AF attempt to cover as wide a range of properties of HCI DETs and design (the things being characterised) as is possible within the bounds of a simple assessment device.
High Generality is addressed, perhaps on too simple a basis, by expanding the view which AF has of the system user. A number of user sub-factors are included in the scoping matrix which allow it to indicate particular human qualities which may be important in certain systems design projects, or HCI DETs. By allowing the design schema and the support matrix to be adapted to suit particular projects it is hoped that AF is not chained to a limited range HCI techniques and design approaches.

A Realistic View of the Design Process is provided by AF in the form of an adaptable schema, in which the components; the information sources, activities and constraints relevant to UI design can be substituted for in instances where appropriate, say when applying the technique to an actual design project. The general form of the schema is, however based upon empirical evidence drawn from two applied and commercial design practice studies.

Coherence of Support AF attempts to support four rather brief roles, not necessarily based in commercial design practice. As a design view, a development aid, or a basis for critiques of HCI approaches to design, it supports a range of practical arguments. If, however, a systems analyst chose to use the framework to convince a manager and design team of the usefulness of one HCI DET over another, it is not clear that the present form would support the types of argument needed. Because of the lack of information provided by HCI DETs themselves on the subject, it is difficult for AF to compensate by providing the kind of information needed, say for a cost benefit analysis as suggested by Mantei and Teory (1988). Were this information available then the scope of a technique should help in suggesting whether the cost of applying a particular technique was worthwhile.

Preservation of Important Information is perhaps the main reason for the existence of AF. It attempts to highlight information which is important to the discipline of HCI as a whole and seriously needed by HCI DET developers, and systems designers trying to choose a technique for their own design. At present too little attention is paid to the information required by designers to choose and successfully apply HCI DETs. Although claims are often made that techniques are indeed applicable to design (Moran 1981; Card et al 1983; Kieras & Polson 1985; Payne & Green 1986), too much attention seems to be focused upon the theoretical defensibility of their methods, and too little on providing this necessary information.
8.7 Summary

AF attempts to fulfill an apparent gap in the field of HCI. There appears to be no coherent, or agreed basis for allowing various techniques to be compared in terms of their practical value to design. HCI DETs do not even present empirically based views of design, or evidence that they do indeed satisfy the claims for applicability made for them. If designers are prepared to adapt techniques themselves, or modify their design approaches to suit the application requirements of HCI DETs then there is no problem for HCI. It is not unreasonable to expect that a new design approach requires some changes in behaviour in design practices. However is the new approach is only able to deal with one part of the system's development then it is not clear how a project should assimilate the new approach, given other design activities and pressures.

The evidence presented in this thesis suggests that systems designers are not assimilating HCI approaches into normal design practice. Even some HCI specialists appear to prefer to use intuition (see Chapter 6). However there is currently a movement in the field of HCI towards making explicit attempts to integrate HCI methodologies into SADMs Wasserman et al (1986) combines prototyping with a structured approach towards user-oriented design; Sutcliffe (1988) exploits existing JSD notational conventions from an HCI perspective, Damodaran and Beck (1988) describe adherence to human factors principles and user-oriented design procedures within combined SSADM (e.g. Downs et al 1988) and project resource management methodologies.

Although such efforts are urgently needed, it does appear from the design studies reported in chapters 3 and 4, that informal approaches to design are more popular than adherence to some SADM. Furthermore, the design of HCI DETs to suit one particular SADM such as a top-down approach like SSADM, may make a technique inapplicable with another bottom-up approach such as "ObjectOry" (Jacobsen 1987), and of course vice verca. Each single approach may only be used on a tiny minority of design projects as a whole. So unless an approach is particularly prevalent in practice, it may be more realistic to attempt to integrate with the most commonly observed approaches, even if they are varied and unpredictable. As Lyytinen (1987) points out, even IS design approaches (another term for SADMS) have weaknesses which make them inapplicable in some cases. Consequently the inapplicability
problem may not be solved by integrating HCI with a theoretically based, but unproven SADM.

AF attempts to compensate for its minimal design-cycle view, which represents commonalities between very different projects, by representing the probable information sources which will be available, and the likely constraints on practice. If an HCI DET reflects the AF design schema, unpredictability in the design process may be less of a handicap because the technique is already designed to cope with the most important problematic constraints which could emerge, and to exploit whatever information is likely to be available, rather than having requirements which cannot be fulfilled in practice.

Over all of the projects examined in this thesis, only two cases involved use of HCI DETs of the type described in Chapter 2, and in these cases the analysts were HCI specialists, unable to carry out user evaluations. This suggests that HCI techniques are not penetrating the commercial market, and are rarely applied in practice.

The four potential roles for AF, a realistic design view, an HCI DET development aid, a critiquing aid, and a selection tool, are all important roles which may help to improve the applicability of future HCI DETs. The selection tool role is perhaps the hardest to fulfil as long as HCI DETs are being developed without consideration for the people who might want to use them, or the design situations within which they could be applied. AF cannot make up for inadequacies in the practical aspects of HCI DETs, it can only highlight them. Therefore, for the time being, the other three roles may be the only important ones.

Unfortunately AF itself fails to satisfy some of its own desirable feature recommendations. It has not been shown to be applicable as a selection tool in practice, although an attempt has been made to demonstrate this role. It may turn out that the present form of AF needs to be developed into something more explicit (perhaps in the form of a manual), but space does not permit such an effort to be attempted here. A good deal of empirical work would be needed, and a more exhaustive characterisation of HCI DETs than provided in Chapter 2 would be necessary since we cannot assume that anyone using AF would have sufficient time to characterise all potentially applicable HCI DETs, before using AF. It is however suggested that this should not be the job of AF, and that HCI DET developers themselves should
provide this information in a more easily digested form than they do at present.

In conclusion, then, AF represents a distillation of the findings from the research reported in this thesis in a potentially useful form which allow them to be put to some practical use in HCI. Further work is needed to develop the framework into something which could be applied in the form of a form of HCI DET selection tool by non-HCI specialists.
Applicability of HCI Techniques to Systems Interface Design

Chapter 9

General Summary and Conclusions

9.1 Overview

This short, concluding chapter summarises each of the preceding chapters, how they contribute to HCI, and how the studies might be improved upon. It also considers some general implications of the research reported for HCI and recommends further research in the area of application of HCI DETs.

9.1.1 Summary of Chapters 1 to 8

Chapter 1 attempted to address the scope of HCI with emphasis on those aspects of particular concern to modelling-oriented HCI techniques. A number of important evaluation factors were identified. These factors represented important perspectives on system properties which must be considered together in order to produce a system which is usable. Evaluating a system from only one perspective could lead to serious misconceptions. Six different usability principles were also identified, each of which reflects some property of a system thought to be influential in determining its usability. In order to illustrate the importance of maintaining a broad perspective on UI design and evaluation in general, some implications of considering these different principles with respect to the various evaluation factors were discussed.

Chapter 2 examined a number of influential HCI design and evaluative techniques. They were classified in terms of whether they were a design or an evaluative approach, whether they represented competence (ideal knowledge representations) or performance (predicting actual user behaviour), and whether they embodied a model of some psychological property of the user, or of model the interactions between the user and the system. Their scope with respect to UI design and evaluation was characterised in terms of a matrix which mapped usability principles onto evaluation factors, each principle and evaluation factor was described in Chapter 1. Each technique was seen to be more or less limited in scope, and as a collection, the
set of techniques reviewed raised some suspicions which were addressed in the following chapters.

Chapter 3 presented a more general discussion of HCI techniques of the type reviewed in Chapter 2 in terms of their apparent applicability to design. It outlined the design views embodied by those techniques which are supplied with explicit statements about the nature of the design process into which they might fit. These design views were criticised and contrasted with alternative user-oriented and empirical views of design practice. Several questions were raised by a number of discrepancies identified between the different design views. These questions were the basis for three studies of UI design as practised by systems designers and HCI specialists.

Chapter 4 described a questionnaire study of the features of UI design practice as it occurs in applied and particularly commercial environments. Nine hypotheses were proposed for the study relating to claims made by the authors of some techniques about their applicability to design. The hypotheses were assertions about the invalidity of various claims which did not seem to have been empirically validated by the authors of techniques. These related to suspicions of the author, informal communications with systems designers, and assumptions implicit in HCI techniques. The questionnaires yielded a great deal of data about approaches to user-oriented design, and various important aspects of the environment within which it takes place, some of which provided evidence to support the hypotheses, and some of which provided valuable information about pragmatic concerns within design projects, such as user-oriented information sources and design constraints which may have an important impact upon the outcome of the project.

Chapter 5 reported on a supplementary design practice study based upon interviews with designers. This investigation attempted to give a more qualitative picture of the design process in terms of sequences of events and how various problems arose for the designers. Systems designers were seen to make no clear separation between the design of the UI and the rest of the system in terms of their attempts to ensure usability. This finding supported the view taken by the scoping matrix used in Chapter 2, that many factors need to be considered together when designing and evaluating with respect to usability. The activities of the designers and the problems they experienced were related to the findings of the previous study, and togeth-
er the questionnaire and interview studies provided the basis for a representative design schema which emphasised three aspects of design practice, as it typically occurs, which are relevant to the application of HCI DETs. These aspects were the goal oriented design activities, the most important information sources exploited, and the most important design constraints which impinge upon practice.

Chapter 6 described a study undertaken to provide information about use of user-oriented and HCI design and evaluation techniques. Since systems designers did not seem to be using recommended HCI approaches, HCI specialists were interviewed to provide a view of the use of such techniques. The specialists in the study concentrated their activities around user-oriented considerations and used a number of techniques to carry out analysis, specification and evaluation. They themselves experienced a number of special problems such as late involvement in design projects, being ignored or not taken seriously, and being expected to spread their efforts thinly over a large number of projects. The techniques which they used were judged by these specialists to have a number of properties which made them more or less applicable in "real world" design projects.

Chapter 7 summarised the findings of the two systems designer studies and attempted to relate these to HCI DETs in order to demonstrate the many obstacles which are likely to make it impossible for many designers to use them. The HCI DETs reviewed in chapter 2 were summarised as having a number of application requirements which are not typically satisfied by design practice. The questions raised in Chapter 3 on the basis of discrepancies between HCI DETs' and other approaches' design views were tackled here. The overall impression seemed to be that a number of fundamental issues relating to applicability of HCI DETs are not being taken seriously by their developers.

Chapter 8 provided an overview of the findings from the HCI specialists interview study. It proposed a number of reasons for why various HCI DETs seem to be applicable or inapplicable; there seem to a variety of general properties which are major determinants of this. These properties were refined into a number of more explicit desirable features for applicable HCI techniques.

In the following part of the chapter a framework was outlined for assessing applicability of HCI DETs. The framework is essentially a distillation of the findings from
the research reported in this thesis and embodies a view of the important aspects of systems design as it relates to HCI. It may be viewed as having a number of possible roles, perhaps as an empirically based design view, a tool for guiding the development of applicable HCI approaches, a basis for critiques of HCI DETs, and as a selection tool for determining an applicable technique for a design project (if the selector has sufficient knowledge of the HCI techniques he or she is considering).

9.1.2 Summary of Contributions of the Research to HCI

A great deal of interesting information about user-oriented design practice has been revealed by the research in this thesis which accounts for its length. Much of the information from the three systems design and HCI practitioner studies may be of use in providing a general picture of the realities and difficulties of ensuring usability in applied and commercial environments. Some more specific contributions made to the science of HCI are listed below.

* A number of user-oriented evaluation factors have been identified which can be viewed as providing a wider context for considering the realism of any HCI assessment of a UI.

* The evaluation factors are mapped onto a number of usability principles in a usability scoping matrix which is a device for representing concisely the scope of any HCI design or evaluative technique (DET). This scoping matrix is used to characterise a number of influential existing HCI DETs and techniques applied by HCI practitioners in commercial environments.

The matrix can be used to characterise the user oriented analytic requirements of an applied design project. It is also expandable and each cell can be assigned a weighting which represents its importance in a technique or to a particular design project.

* The design views of a number of HCI DETs are made explicit and contrasted with alternative views of design practice. Two studies of user-oriented systems design practice as it occurs in applied and commercial environments demonstrate two points. Firstly, HCI DETs of the type reviewed in this thesis are rarely used, if at
all, by non-HCI specialist systems designers and HCI specialists are a rarity in commercial design projects. Secondly, there are a number of practical reasons as to why HCI DETs are inapplicable. These reasons are made explicit in chapter 7 where the application requirements of HCI DETs are contrasted with the realities and constraints of applied and commercial design practice.

* The two user-oriented systems design studies reported in chapters 4 and 5 combine to provide a simple empirically based design schema which represents the goal-oriented activities common to various design projects studied, together with the ten most common user-oriented information sources used, and the ten most common design constraints which were reported in Chapter 4. This design schema provides a more realistic and practical view of design as it occurs in practice, together with its problems, than those idealised views embodied in HCI DETs.

* The HCI practitioner interviews yielded further information which, in the light of the previous two studies, led to the identification of a number of specific desirable features which a generally applicable HCI DET should have. These features are incorporated within a framework for assessing the applicability of HCI DETs (AF). This framework embodies the empirical findings of this thesis and presents them in a form which may be used to fulfil four roles, which are as follows;

  * The Design Schema presents a realistic picture of the type of situation an applicable HCI DET should be designed to deal with.

  * The Scoping Matrix presents a space within which any particular design project or HCI DET can be characterised. In the former case it would represent areas that required special analytical attention (see the example in Chapter 7), and in the latter it would represent areas which a particular technique is capable of addressing (as in chapters 2 and 7).

  * The HCI Roles and Activities Matrix suggests coherent support for HCI roles which an HCI DET might attempt to follow.

  * The Desirable Features List makes explicit a set of potentially evaluable features which are desirable for an applicable HCI DET. These features are based upon the findings from the three design studies reported
in this thesis.

9.1.3 Shortcomings of the Research

The HCI DETs reviewed in this thesis are only a small subset of the techniques which are currently in existence and there is some risk that they might be unrepresentative. They were chosen on the basis of their being influential and well known within the HCI community, however other techniques which are also influential have not been considered and may not suffer from the same weaknesses.

Both the questionnaire design and the interview design for the two user-oriented systems design studies were aimed at gathering as much information as possible without preconceptions about what kind of information was to be revealed (as a result the questionnaires were over long and took a great deal of time to complete which may account for the fact that only 18% were returned). For this reason some interesting features of design practice were only hinted at accidentally by participants in the studies. Further, more directed studies along the same line may be necessary in order to pursue such information.

The relative rarity of HCI specialists in commercial organisations made it difficult to obtain volunteers for interview. Consequently only a small number were included in the HCI practitioner study. Luckily the participants came from four different organisations with markedly different working regimes, providing a wide range of approaches to HCI in systems design. However this set may still be unrepresentative of HCI practitioners as a whole.

There was insufficient time to attempt a real analysis of the utility of the application framework in any of its potential roles in Chapter 8. Consequently its most demanding role (that of an HCI DET selection device for commercial designers) had to be demonstrated using one of the design projects from Chapter 5 as an example. For this reason it may have many practical shortcomings which will need to be ascertained and dealt with before it can be presented as a widely accessible HCI tool.
9.2 Summary of Implications for HCI DETs

A great deal of information has been presented in this thesis and many interesting facts about user-oriented design as it relates to HCI techniques have been revealed. It remains however to pull out the major implications for existing or future HCI techniques of the type described in Chapter 2, since the evidence seems to suggest that HCI runs the risk of being marginalised within systems design practice, where many other considerations compete for the attention of the design team. In the following discussions a number of newspaper-style headlines are presented and discussed which sum up the important findings emerging from this thesis.

**Power versus Pragmatics: Where Should the Emphasis Lie?**

Descriptive, explanatory, or predictive power seem to be the main obsessions of developers of HCI DETs in terms of how they demonstrate their value. Detailed laboratory studies are used to vindicate the accuracy and sensitivity of various techniques (e.g. Card et al, 1983; Kieras & Polson, 1985), or lengthy and precise specifications of existing or imaginary systems are presented (e.g. Moran, 1981; Payne & Green 1986) in order to vindicate the theoretically based claims for their power in some respect. The impression gained by the reader is that the techniques should be applied in the same way as they are demonstrated. However it is obvious that the demonstrations reflect contrived circumstances (such as laboratory conditions, or analysis of a completed or imaginary or very simple system) and the applications chosen in these circumstances are perhaps selected because they do not raise complex problems outside the scope of these techniques (e.g. text editors, "email" systems, and drawing tools).

It is hardly surprising then that the impressive accuracy of predictions, or the sophistication of explanations of usability problems do not cut much ice with the analyst in the "real world", particularly if he or she is not an HCI specialist. A general feeling beginning to emerge amongst the HCI population is that generality of applicability is more important than precision (e.g. Barnard 1986), and that, in the design or evaluation of a UI, adherence to the spirit of a technique may bestow almost as much value in terms of ensuring usability, as applying the whole technique in all its detail (Maclean 1989, personal communication).
9.1.4 Related Work on Scoping of HCI DETs

The AF scoping of HCI DETs for the purpose of examining their analytic breadth may be compared with two other approaches which have similar aims. These are Simon's Trade-Off approach (Simon 1988) and Young and Barnard's Scenario approach (Young & Barnard, 1987; Young et al, 1989).

Tony Simon has identified the problem of the difficulty analysts must have in identifying an evaluative HCI modelling approach which best suits their requirements, given that no model is able to tackle all design issues at present. He presents a multi-dimensional representational space, within which HCI modelling techniques, such as those reviewed in Chapter 2, can be characterised. They are portrayed as three dimensional shapes in different shades which represent the functions for which they were intended or the output they produce. Their depth in the space represents scope of the processing resources (motor, cognitive, and/or sensory) which they capture.

Their position on the vertical axis illustrates the extent to which idealisation of behaviour being modelled (to produce concrete predictions) is traded off against the qualitative aspects of the mental operations which support behaviour (allowing greater generalisation of the model). The position of the models on the horizontal axis also relates to where they stand on a trade-off. This is between the extent to which a model enumerates the data which is mentally manipulated, and the extent to which it identifies what knowledge is exploited in mental processes for various tasks. Another trade-off identified by Simon is that of input-output whereby models which require much effort to build, must justify this effort in terms of greater value of output. This recommendation has been referred to in previous chapters in this thesis.

What Simon's work seems to do is present a characterisation of HCI modelling techniques in terms of analytic features and trade-offs which are qualities of the models themselves. This work does not refer to the actual UI properties and factors which temper evaluation as does the AF scoping matrix. It is therefore up to the analyst to intuit from Simon's characterisation whether any given model would be appropriate for a design question. The scoping matrix is more clear in this respect, and the AF covers practical design application issues which Simon's work does not
A more similar approach to that of the matrix is the work of Young and Barnard (1987) and Young et al (1989) which uses sets of behavioural scenarios which cover the same type of ground addressed by the scoping matrix. The behavioural scenarios are like instantiations of configurations of cells in the scoping matrix. A behavioural scenario is a brief characterisation of some robust phenomenon of interaction between a user and a computer. An example would be the user typing an abbreviated command "d" to move down in a text editor, when in fact the effect of this command is to delete the previous character. Such a phenomenon occurs in many circumstances and raises a number of issues for modelling techniques which could be characterised in the scoping matrix (perhaps as appearing in the Compatibility column on the User, UI and Target Tasks rows).

Collections of scenarios presenting different aspects of behaviour may be used to represent the scope of issues a modeller wishes to be able to model. Aspects of the scenarios which a model fails to address represent the limits to its scope, as do empty cells in the matrix. It would probably take a great many scenarios to fill the entire scope of the matrix. Many scenarios will tend to overlap. Therefore the matrix is perhaps a convenient abstraction which is capable of summing up more clearly the scope of a given technique.

One of the weaknesses of the scoping matrix is that it does not currently have devices to represent the grain of analysis taken by an approach to a given area of interaction. Nielsen (1986) has usefully compared a number of approaches to dialogue characterisation to show the extent to which they are capable of capturing levels at which dialogue may be said to occur. Some levels, such as the Syntactic and Lexical Levels of CLG are clearly observable. Others such as the CLG Task and Semantic Levels may only be inferred, and are consequently more difficult to model with psychological plausibility. Since different units of information are conveyed at different levels of dialogue, some units such as task goals may only addressed at a high level of abstraction, others such as keystroke sequences must be addressed at a low level. It would be useful to capture this variation within the usability scoping matrix, but at present this has not been addressed.
The scepticism about HCI DETs apparent in systems designers contrasts with their generally positive attitudes to user-oriented design. By suggesting that they use highly complex, unfamiliar, and esoteric techniques, which do not seem to allow for a rough approximation, HCI technique developers may create the impression that HCI is always difficult to apply, and they may be providing an excuse for designers to ignore HCI altogether.

What seems to be needed is some trade off between considerations for the overworked under-resourced systems designer, and the scientific defensibility of theoretically based techniques. It may be that making techniques easy to apply means that they will also have to be made less precise. However, instead of evaluating techniques for their psychological validity and analytic power we should be evaluating them in terms of the sense, and practical use, that can be made of them by the non-HCI specialist.

The Price is Right:
Practical Cost Benefit Considerations for HCI

The findings from the two systems design studies suggested that inadequate time and resources are a major problem for systems design projects. Yet cost benefit considerations are given scant attention by the HCI community, with the assumption being that any technique which can demonstrably predict the effect of some aspect of the UI in terms of user performance must be valuable. Management in systems design companies are less likely to risk running over budget by spending more on ensuring usability, or any other property of their product, if they have no idea what commercial benefits they will accrue by doing so.

HCI must take a more serious attitude to the requirement for demonstrations of the real commercial value of applying HCI techniques (e.g. Mantei & Teory 1988). At present the cost of applying some technique versus proven economic returns is not considered in anything more than a hypothetical sense by the developers of HCI techniques. This may be a problem of the nature of HCI research which seems to take place in something of a vacuum, rather than within some design project. Gould et al (1987) are an exception to this rule as they were able to provide some demonstration of concrete beneficial effects of adhering to a user-oriented approach in practice in an applied project, even if they were not able to prove that the precise cost of
not applying their approach would have been greater.

The "Moral High Ground": Where does HCI Stand
With Respect to User-Oriented Design and Evaluation

It seems to be a popular notion that systems designers willfully go about designing unusable systems without paying any attention to their prospective users. This thesis has attempted to suggest that they are not entirely to blame for this. Pragmatic considerations such as the difficulty of making late modifications to software, inaccessibility of prospective users, and so on tend to make the designers job far from easy. They are therefore not helped by being exhorted to take up HCI techniques which make few if any concessions to the difficulty of design practice, and it is unreasonable to expect them to redesign such techniques themselves.

Where HCI DETs were observed to be used in applied or commercial projects, they were used by HCI specialists who were able to concentrate their efforts on the UI, and had the skills and experience to adapt the techniques, where necessary, to their requirements. However HCI specialists are present in only a minority of design projects, and it seems that even they find that HCI techniques need modification. In general it seems more reasonable to suggest that the onus is on HCI DET developers to aim for greater comprehensibility of their techniques as well as for theoretical accuracy, such that systems designers can use them successfully, if they wish to see their approach in use in applied and commercial design projects.

Mohammed and the Mountain: HCI and The Rest of Design

HCI research has successfully demonstrated many inadequacies in existing UIs and calls upon designers to change their practice and develop less system-centric approaches. The interview study reported in Chapter 5 strongly suggested that systems designers have many considerations to bear in mind, and that their responsibility to the user is really only one of many since they have employers budgets, market forces, and software considerations as well.

The features analysis and the interview study of design practice suggested that design approaches, though highly varied, are more likely to be informal, and to involve iterative prototyping with user-evaluations, than they are to be top-down
structured specification-oriented. Whatever the case, the limited notions of the system life-cycle embodied in HCI DETs, which may be inadequate, certainly do not suggest how they should be related to other aspects of the design and competing considerations important to the ultimate character of the UI such as in-house style, software and hardware limitations, inherent complexity of target tasks and scoping of system support for these.

Greater effort therefore is required in order to make HCI techniques self sufficient (i.e. not dependent upon any possibly unfounded assumptions about the design approach) or easy to integrate with other approaches which deal with the considerations which they ignore. It is unlikely that the mountain of design practice will come to the prophet of HCI if as it seems, HCI is so poorly represented at present (see Chapter 4). So the discipline of HCI must address this problem and begin to turn out techniques which will fit in with current practice.

The Application Framework (AF) described in Chapter 8 set out to address this issue. It attempted to represent design in terms of its goal directed activities, typical user-oriented information sources exploited, and typical design constraints. It lays out the scope of a technique, in such a way that it can be compared with other techniques, or with the requirements of a design project. It attempts to characterise coherent HCI analyst roles which involve various types of activity which might require support. Finally these components are used to help address an ideal features list which stipulates a number of specific features which should improve the applicability of an HCI DET. At present the main obstacle to the use of this framework within design projects as an HCI DET selection tool is the frequent lack of explicitness amongst the HCI techniques about their design process assumptions or their scope, and suitability for different application domains.

HCI: Who Needs it Anyway?

The features analysis of of user-oriented design practice suggested that HCI specialists tended to be employed on large design projects. Mantei and Teory (1988) suggest that projects over a certain size are more likely to profit from the application of HCI than are small projects (but do not base their analysis on actual application of various HCI techniques). The features analysis certainly seemed to confirm that this may be commonly believed to be true by those who determine when it is appropri-
ate to invest in the services of an HCI specialist. However, it seems irrational to suggest that it is the nature of the design team size, or project length which should dictate whether or not HCI is worth applying. This is something which should be addressed by HCI researchers; that there may be a strong requirement for HCI techniques of some sort on small, less well resourced projects.

Something resembling a market analysis could identify the type of system which would be suitable for certain types of HCI DET. Some systems, such as those in high risk domains like power generation must be designed to avoid human error, others like automatic telling machines should be designed in such a way that everyone will be able to use them; these types of system have a strong requirement for HCI input into their design. Other systems, such as text editors (which seem to be a popular target for demonstrations of HCI DETs) may not require much HCI input into their design in order that they be marketable; in fact they may be designed by following styles and ideas manifested in other comparable systems (see Chapters 4 and 5).

The requirements of small and large projects designing varied systems, and the type of analysis they can support are bound to be diverse. Perhaps HCI DETs could get away without being generalizable to all UIs if they were targeted at some identifiable class of design project which could clearly benefit from their input. A technique aimed at designs where HCI is not the most important concern, should be cheap and easy to apply. An evaluative technique should offer the most appropriate usability or performance metric for evaluations; for example it is not clear that speed of ideal users is the most appropriate metric for all text editors (GOMS; Card et al, 1983, for example, concentrates on this metric in evaluations of text editors).

Some considerations for designers were suggested in Chapter 1, however it was not clear that any of the HCI DETs described in this thesis were targeted in a coherent manner towards classes of applications other than those like text editors, involving simple user tasks, no real-time system output behaviour, and focusing on limited user psychological characteristics. HCI DETs should explicitly describe the application domains for which they are appropriate, the user populations best simulated by any model they present; e.g., naive users, expert users, domain experts and so on. They should emphasise whether they focus on perfect performance, speed, error prediction, learning and so forth whilst being explicit about any assumptions being
made about users; e.g. the training to which they have been exposed, their representativeness of some population, the conditions under which they are expected to interact with the system (e.g. when they are not tired or frightened), their tasks, and so on.

The discussion about who needs HCI appeals for detailed breakdown of the scope of a technique to be carried out by its developers. Different design projects have different requirements for analysis. The usability scoping matrix applied in in Chapter 8 would be a suitable guide to the effort of providing a scoping breakdown of a technique. Each cell addressed by a technique should be considered in precise detail, with assumptions about each of the evaluation factors being made explicit. Ensuring that techniques are clearly characterised by their authors should make them easier to select by systems designers.

9.3 Recommended Further Research

In this chapter several suggestions have already been made for further research in HCI which should enhance its applicability. These are briefly summed up here.

**Proven Benefits From Use of Techniques** need to be demonstrated before making claims about the utility of a technique for improving usability. Demonstrations by the developers themselves in contrived circumstances, on existing systems are unlikely to convince systems analysts and engineers that a technique has any relevance to real world design practice.

**Integration** with other design activities should be a prime concern for HCI DETs which do not, in themselves, represent complete methodologies. It has been suggested frequently in this thesis that current techniques of the type described in Chapter 2, do not seem to be well adapted for use in a real design project; they have various application requirements which often do not appear to be satisfied by applied and commercial design.

**Explicit** scope and procedures in techniques need to be presented in order for systems engineers to be able to select the one which best fits their analytic needs, and which is best suited to what their project can support.
**HCI Techniques Should be Targeted** at particular application domains or types of user and task with which they are best suited to deal. If they are not targeted, then they should be highly generalizable (thereby running the risk of being complex to apply). Any targets should be realistic, reflecting typical or common contexts of design projects rather than reflecting convenient assumptions made by the developers of the technique.

**Non Esoteric** techniques are required since most design projects do not involve HCI specialists with the skills and experience to understand and modify a technique where necessary. There seems to be a greater emphasis on accuracy of predictions, rather than ease of use of HCI DETs. Either a trade off will have to be made between accuracy and ease of application (since sophisticated models are complex and time consuming to build), or future techniques will need to be embodied within automated user models or support tools which make the task of analysis of, or design for usability much simpler for the non-HCI specialist.

**The Application Framework** needs to be empirically validated as a potential HCI DET or user-oriented approach selection tool. At present it requires that the analyst using the technique would require considerable familiarity with each of the approaches he or she was considering. Either a taxonomy of a wide variety of existing HCI DETs should be supplied with the framework, or (more desirably) HCI DET developers should be more explicit about the scope, limitations and assumptions of their technique.
References


Barnard P. (1985), "Interacting Cognitive Subsystems: A Psycholinguistic Ap-


Kosslyn S.M. (1985), "The Medium and the Message in Mental Imagery: A


Mantei M. & Teory (1988), "Cost Benefit Analysis for Incorporating Human Fac-


Books.


**Definition:** The Multiview Approach, Blackwell Science Publications.


Appendix 1

Questionnaire Distributed to Systems Designers in Features Analysis of Applied and Commercial Design Practice

Please return copies to me:

Victoria Bellotti at

Queen Mary College
Department of Computer Science and Statistics
Mile End Road
London E1 4NS

tel: 01 975 5200
messages only

home: 01 514 8849
I should be very Grateful if you could assist me in my research by filling in this questionnaire. I am very interested in interactive system interface design, but have found it hard to meet and interview commercial designers. This questionnaire is designed to find out about the way in which designers go about collecting the information they need to make interface design decisions. I am particularly interested in commercial design projects, so if you have been involved in a commercial project I should be grateful if you could relate your answers to that in preference to research or other types of projects. However, if you have not been involved in commercial work, then please refer to any other type of project.
Please read the following before attempting to answer my questions.

This questionnaire is aimed at collecting information about how designers go about designing the user-interface to computer systems in commercial projects. I should therefore be grateful if you would base your answers on a single project rather than on your general practice. The first part of this questionnaire is designed to gather background information about the nature of the project and how it proceeded. The second part, which may require you to duplicate some of the information from part I, asks more specific questions about problems, practices and results. It is therefore important that only information relating to a single project is included, otherwise the information I receive will be misleading. Please try to answer all of the questions. Also, if you wish to learn about my overall findings, please write an address below to which I can send the information when I have completed my analysis.

Part I

I need a brief description (approx 2 or 3 sentences, more if you wish) under each of the headings below. Please read all of the headings in Part I before beginning in order to avoid duplicating information.

Product Function(s)
Type of Interface (e.g. WYSIWYG)

Prospective Users

Description of Own Organisation (no name required)

Description of Client/Marketing Organisation (if other than own)

Extent of Involvement of Client and Any Other Organisations or Individuals

Design-Team Size, Roles and Structure (including consultants)

Your Job Description

Your Particular Role in the Design of the Interface

Human Resources (programmers, secretaries, associated researchers etc)
Technological Resources (hardware, software etc)

Time Resources (e.g. available person-hours, deadlines)

Availability and use of Specification Information

Availability and use of Information About To-Be-Supported Activities

Availability and use of User Population Information

Design-Team Familiarity with System Application Domain

Use of Abstract Design Specifications (e.g. Task Analyses, Interaction Grammars etc)

Particular Descriptive Methodologies Adopted (e.g. JSD, GOMS, LARCH)
Give Reasons

Generation and Testing (formal, informal, iterative, etc)
Prototype-Interface Evaluation (self -, prospective-user -, experimentally - tested, etc)

Notable Problems (state when they emerged and how they might have been avoided)

Notable Modifications (state when made and why)

How the Product Interface Was Finalised

Was the Product Interface Satisfactory, if Not Say Why

Rough Initiation and Termination Dates of Project

What You Would Have Done Differently with the Benefit of Hindsight

Do You Know HCI Well as a Discipline, Could it be Useful, (please state why/why not)

Any Additional Information You Consider Relevant
1a). The following is a list of possible constraints on design activities. Please rank any which you experienced in descending order of importance (i.e. 1 is most important). Feel free to make a) to u) more specific if necessary, by adding to them.

a) Lack of Autonomy From Parties Outside of Design Team
b) Lack of Guidance From Parties Outside of Team
c) Lack of Authority
d) Oversized Team
e) Undersized Team
f) Undefined Team Member Roles
g) Over-rigid Team Member Roles
h) Lack of Assistance/Collaboration from Client
i) Client Over-Intervention
j) Lack of Information about Tasks
k) Lack of Information about Users
l) Over-casual Approach to Design
m) Over-rigid Approach to Design
n) Over-casual Approach to Evaluation
o) Over-Rigid Approach to Evaluation
p) Lack of Experience With HCI
q) Lack of Experience with Interface Design
r) Lack of Information About What Constitutes Interface Design Improvement
s) Lack of Familiarity of Application Domain
t) Complicated Application/ Sophistication of Product
u) Inadequate Resources (e.g. Time, Money, Equipment etc)
Rank

Please refer to each item above by its reference letter

1. __________
2. __________
3. __________
4. __________
5. __________
6. __________
7. __________
8. __________
9. __________
10. __________

1b). Any constraints you experienced which are not included above may be ranked below. To indicate where they would fit in use a sub-index (i.e. 0.1 Disorganisation, 3.1 Sickness, 3.2 War etc)

Rank

___________

___________

___________

___________

___________
2a). Please read the list of possible information sources and rank those to which you had access in descending order of importance as in question 1a. Feel free to make a) to r) more specific if necessary, by adding to them. N.B. *The word activity is used to denote tasks, games, learning, etc*

a) Scientific/Psychological References on Human Behaviour etc  
b) Psychological or HCI Task Analyses of Related Activities  
c) Surveys/Reports on Human Characteristics  
d) Documentation on Related Activities (e.g. teaching material or manuals)  
e) Surveys/Reports on Target User-Groups  
f) Specifications of To-Be-Supported Activity  
g) Interviews with Non-Prospective-Users about User Group Characteristics  
h) Interviews with Prospective-Users about their Characteristics  
i) Verbal Task Descriptions from Current Activity Performers  
j) Verbal Task Descriptions from Other Persons  
k) Observation of Prospective User Activity  
l) Observation of Non-Prospective-User Activity  
m) Observation of Activity Independent of System Prototype Use  
n) Observation of Activity Using Prototype  
o) Experimentation with/Testing of Prospective-Users  
p) Experimentation with/Testing of Non-Prospective-Users  
q) Experimentation on Activity with Prototype  
r) Experimentation on Activity without Prototype (e.g. with mock-up)
Rank

Please refer to each item above by its reference letter

1.__________
2.__________
3.__________
4.__________
5.__________
6.__________
7.__________
8.__________
9.__________
10.__________

2b). Any information sources you used which are not included above may be ranked below in the same manner as in question 1b.

Rank

__________
__________
__________
__________
__________
3. Please describe briefly (2 to 3 sentences, more if you wish) how you went about exploiting the information sources you had access to. It would be helpful if you could describe them in the order of importance, as you have ranked them and refer to them according to their ranks (e.g. 0.1, 1, 2, 3, 3.1, 3.2, etc)

Rank
4. Please describe the degree to which you are satisfied with the user-interface as it was after finalisation. If possible, it would be very helpful if you could include a list of good, and of bad features about the interface, together with reasons. Write on the back of this page if you run out of room.
5. Finally, please read the following list of possible relevant aspects of your interface which are of particular interest to me. Next to any aspects which you tried to incorporate into your interface please write 'Yes' if you succeeded to reach a satisfactory design solution, and 'No' if you did not. Please feel free to give reasons and to extend the list.

Many thanks for your time.

Possible Aspects of Your interface

Ease of Learning

High Consistency/Predictability

Low User Confusion

Minimum Interaction Rules

Few Interaction Steps Per Command

Low Error Rates

Good Error Handling

Good Active Help Facilities

Good Passive Help Facilities

Good Manuals

Good Teaching Material

High Discoverability of Existing Functionality
Modelessness

Well Formed Metaphor

Good Special Key Functions

Easy Use of Mouse

Appropriate Special Input Devices

Good Overall Screen Layout

Good Text Layout

Well Designed Windows

Good Use of Graphics

WYSIWYG

Good Icons

Appropriate Commands

Well Designed Menus

Good Direct Manipulation Tools

Appropriate Colour Coding

Appropriate Functionality

High Reliability
Appendix 2

Systems Designers’ Descriptions of Their Exploitation of Information Sources in Features Analysis of Applied and Commercial Design Practice

Exploitation of Information Sources

Where rankings have been supplied by the respondents these are shown in the tables. Additional comments, where ranked are described in order according to their relative importance with respect to other information sources. The information sources are referred to by the letter associated with them in the following list.

a) Scientific/Psychological References on Human Behaviour etc  
b) Psychological or HCI Task Analyses of Related Activities  
c) Surveys/Reports on Human Characteristics  
d) Documentation on Related Activities (e.g. teaching material or manuals)  
e) Surveys/Reports on Target User-Groups  
f) Specifications of To-Be-Supported Activity  
g) Interviews with Non-Prospective-Users about User Group Characteristics  
h) Interviews with Prospective-Users about their Characteristics  
i) Verbal Task Descriptions from Current Activity Performers  
j) Verbal Task Descriptions from Other Persons  
k) Observation of Prospective User Activity  
l) Observation of Non-Prospective-User Activity  
m) Observation of Activity Independent of System Prototype Use  
n) Observation of Activity Using Prototype  
o) Experimentation with/Testing of Prospective-Users  
p) Experimentation with/Testing of Non-Prospective-Users  
q) Experimentation on Activity with Prototype  
r) Experimentation on Activity without Prototype (e.g. with mock-up)
The Design team collected information **directly** from potential users. Used paper tools (Kelly's personal construct theory) to represent system view and checked these with potential users.

e Designers used "this kind of thing" as **initial input** before respondent joined the team.
f "Functionality & tick lists" provided by marketing early on "and just filter into the design."

**o, p, q, r** All the outputs used similarly: difficulties agreed, solutions proposed, cost/benefit analysed, changes made.

i, j "Talking about the task to be performed led almost automatically to the chosen design. The users had already divided the task into meaningful subtasks."

**d, k, m** "Manual systems were highly developed for the task and provided further structure to the task. I didn't try to draw the charts on the screen, rather to produce printouts that correspond to the manual summary books."

h "It became clear that the operators were not allowed to alter the system, but that they had considerable discretion over how to organise their work, e.g. shifts rarely balance exactly, so the operator has the final word on accepting/rejecting shift figures."
The system however ensures that all sections have been entered before reporting the discrepancy and sections can be corrected individually, without extensive retyping.

The designer did get to see the system in daily use "I shared an office with the operator", but it needed only trivial changes to increase the speed of keying.

5A

<table>
<thead>
<tr>
<th>Source</th>
<th>a</th>
<th>n</th>
<th>p</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rank</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

The design team used people in the company and text from the CHI'84 workshop.

"Even if we had wanted to, there was not enough time to look for further information sources. It was not clear where there would be any (we were putting together a novel system, and did not know who the user base would be)."

6A

<table>
<thead>
<tr>
<th>Source</th>
<th>d</th>
<th>f</th>
<th>j</th>
<th>k</th>
<th>l</th>
<th>m</th>
<th>a</th>
<th>n</th>
<th>h</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rank</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source</th>
<th>c</th>
<th>q</th>
<th>o</th>
<th>p</th>
<th>e</th>
<th>b</th>
<th>g</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rank</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
</tr>
</tbody>
</table>

d Searching the related literature in professional publications. Reading a competitor's documentation. Trying to identify pitfalls, drawbacks, etc from the documented description.

f Respondent attended in house short course on chip design. Attended exhibitions, demo’s etc of competitors. Collected information from the professional press, and investigated existing tools.

j Discussions with management.

k Observations of the in-house electronics engineers using existing tools. Discussions with them getting subjective evaluations of existing tools, and eliciting requirements mainly based on drawbacks of existing tools.

l Observation of in-house software engineers using other tools, observing the
learnability process.
m Included in k.
a, b, c, e Searching available literature.
n Tools released for engineers to test and respond to at various stages in the design process. "The common response was that they ignored the release altogether due to the bugs which were still in it, and the low level of reliability of the tool, and would carry on using existing tools."
h Informal discussions were held with engineers and layout designers. They were useful for identifying deficiencies in existing tools, but not as useful in presenting a systematic and comprehensive set of requirements. "They were better at reacting than initiating"  
i As in h. 
q Prototype did not stay in-house for experimentation, Various tools were constantly released in-house. However, the fully integrated prototype was sent out for prospective customers for a three months Beta-testing period. 
o No experimentation as such. Observations of use of externally produced existing tools and various system versions, as it progressed. 
p Very minimal in-house experiments on software people "Usually to resolve HCI arguments when no side was giving in."

7A

<table>
<thead>
<tr>
<th>Source</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>k</td>
<td>3</td>
</tr>
</tbody>
</table>

The only information source was a design team member "using his immediate experience and his observations for the need of the product and the prospective users. Once the prototype was working, he tested it and then let prospective users use it. It was then modified and improved." The other design team member used experience with other products and own design. "We have not researched the design interface or the application, we have used experience and watched the users."
The designer has had experience of many products this led to the UI being designed through discussions. Existing products were examined to determine how they behaved.

10A

<table>
<thead>
<tr>
<th>Source</th>
<th>f</th>
<th>q</th>
<th>a</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rank</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

| Specification of user activity was very global in nature; i.e. ALL user activity would occur via our interface. |
| Technical papers Al\ algorithms and overall design was specified using the "technical references" available. |
| Feedback from prototypes helped the team to "fine tune" many aspects of the system. |
| Use of reaction time information, and information on colour sensitivity. "Would have liked info on interpretation of colour and symbols." |
| Same as for q. |

11A

<table>
<thead>
<tr>
<th>Source</th>
<th>d</th>
<th>a</th>
<th>n</th>
<th>k</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rank</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

| Hardware Specifications These were essential to control machine resources. |
| Papers on Graphics Used to develop graphics algorithms. |
| Papers on Colour Used to develop algorithms to change between colour models. Used to develop WYSIWYG colour editors. |
n Allowed (the potential users) to play with self contained modules of the program to see where they became confused or managed to disrupt the program. Then modified the UI and decided on the order of presentation of the modules.

g Consulted with users' supervisors as to whether users would work alone or in groups, and how much it would be possible to leave them alone; i.e. what level of sophistication they would be able to deal with unaided.

j A large number of modules was needed. Some were contributed from outside the design team.

h When users had used the programs under the observation of their supervisors both were encouraged to suggest problems and improvements.

e, d These were provided by the supervisors' advisor and improvements were made in consultation with him; e.g. alterations to vocabulary used on screen/difficulty of certain aspects of the program.

k, o Users were allowed to use the computers apparently unattended, and their use of the keyboard noted. Tests were performed to test which types of display were most noticeable/ most readable etc.

a, c Not used much, but helpful in fine tuning the delays & and responses of the system to make it feel as natural as possible.

There were two responses from this project, however only one gave elaborations on how information sources were exploited.

User requirements analysis on document preparation (the application domain) was very important. Opened up a lot of possible areas for adaption; used "rich pictures" to show the user requirements analysis on each person.
From it was possible to identify a product which could incorporate various areas of adaption and thus provide a design/chart-of-areas-to-be-researched.

3B

<table>
<thead>
<tr>
<th>Source</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rank</td>
<td>1</td>
</tr>
</tbody>
</table>

Experience with "spy" "Spy set a standard that I sought to equal using techniques that assured the user that he/she knew what was going on. I sought to use some of the principles used in spy in my own work."

Experience with previous UI product design "On an earlier project I had the opportunity to work directly with users of the package I was building, and respond directly to feedback I received. This was a useful experience."

The package used "enables a good UI to be constructed in a remarkably short time... has some annoying features and the original manual was atrocious, but it provides a very productive environment, with a resulting interface that I can be proud of."

Demo program of the package used Because of weaknesses in the package used "demo programs were essential in getting me on the right track in the beginning."

4B

<table>
<thead>
<tr>
<th>Source</th>
<th>n</th>
<th>h</th>
<th>o</th>
<th>q</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rank</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Reference Manual of the graphical toolkit used. Source code of existing applications which shared features with the new application. Systems manuals.
Existing interactive graphics modelling programs and literature relating thereto. Existing demo programs for workstations. Computer manufacturers literature and manuals.

"We wrote the usability spec. We researched related activities and different interfaces.

"Client visited us to see some early versions; some useful feedback. We used our own internal reviews as we had a simulated system to interface to."

m, k Design team very familiar with application domain; used intuition to determine the most helpful UI.

f "Design team member wrote a document describing formally the model of
theorem proving (the application domain) we would support. He based it on papers by mathematicians [application domain experts]."

d "There have been a number of systems like ours, though with not so good HCl. We did a survey of these."

i, g "Informal commentary from mathematicians [application domain experts] during our design conferences."

n, q "We built a prototype system, tried it out ourselves and showed it around a lot."

1C

<table>
<thead>
<tr>
<th>Source</th>
<th>h</th>
<th>f</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rank</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Talking to the estimators (application domain expert potential users), understanding what went in to building up an estimate. Did not talk to entry clerks (non-application domain expert potential users) who would have to use the system.

2C

<table>
<thead>
<tr>
<th>Source</th>
<th>f</th>
<th>d</th>
<th>p</th>
<th>o</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rank</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Technical support; IMS Experts Talked to company experts when advice needed.

f "Documentation; read manuals/standards documents."

d "Specification; developed specifications and used them as guidelines."

p, o "Testing with users and non-users; used to check response times and other problems and acceptability to user department."
i Design team reviewed client’s written job descriptions.
h Interviewed clients to determine how the job was currently done.
k Observed clients in workplace.
o Tested UI design (e.g. screens) with non-clients.
p Tested screens with clients.
Appendix 3

Systems Designers' Descriptions of Their Satisfaction with UI Product of Design in Features Analysis of Applied and Commercial Design Practice

The respondents' views of their UIs are shown here. Some have included good and bad features. These responses are not assumed to reflect objective features of the UI, merely to indicate what designers typically regard as good and bad features of UIs which they themselves have designed.

1A It was appropriate because "it struck an acceptable balance between a user and system driven dialogue."

3A Due to extensive usability tests and problems found, designers are going on to implement extensive changes [project not completed].

**Bad Features**: UI is complex and difficult to use and remember.

Mechanisms for search tasks [crucial to the filing and retrieval application] slow and clumsy.

4A Respondent thinks the UI is very good.

**Good Features**: Uniformity. Menus approach to all screens, two formats used consistently for (a) select subtask and (b) action within subtask. "Esc" used for return to previous menu. "Fixed meanings" for navigation keys, including skip back.

No stupid questions and relevant menus appearing only when valid.
Interface division matches task division.

Streamlined, e.g. menu actions only require one keystroke.

Configuration file ensures that data collected exactly matches local requirements, users' locally chosen names etc.

**Bad Features:** Bad hardware

Configuration aspects not flexible enough.

5A

**Good Features:** Consistency of concepts throughout. Careful with names and relationships.

Use of window types as modes.

Consistent use of function keys/menu items between modes.

Command key meanings displayed - non-valid ones are dimmed.

Splitting of screen into areas with different meanings.

**Bad Features:** Technological restrictions of the display to PC text/graphics characters (not a real window system)

Amount of mode swapping necessary to complete the task (even though structure of modes reflects the parts of the task).

6A

**Bad Features:** "The use of this system was quite difficult to master. It could even be said that the mouse buttons were overloaded with functionality which wasn't necessarily 'common sensible' or easy to explore."
"System-wide uniformity was not always maintained, usually as a result of implementation constraints. The end result was that there were too many exceptions to each rule."

"Response time of system wasn't really satisfactory, with its worst effect on direct manipulation, pop-up menus etc."

7A The respondent was satisfied with the UI.

**Good Features:** "Simple uncluttered screens."

"Good use of colour, though not too flash."

"Fast display."

"Pop-up windows to select options. The text in windows is easily configured outside the program."

"This configuration means that the users only see the options applicable to each of them. Hence they are not confused by unusual options."

"Good use of function keys."

"Standardisation of function keys wherever possible."

"Very few key presses to make selections etc."

"Good use of sound."

**Bad Features:** "Function key descriptions are not always descriptive enough."

"Program does not always show order that the function keys should be used in."

"Tried to avoid hidden key presses (screen mostly displays which
keys are currently available). However some key presses are not displayed."

8A "Although the UI is easy to use and looks 'good', there are now on site a proliferation of screen driven systems, all accessible from any terminal. These systems run on various types of hardware. Some are bespoke and many are packages. It would be helpful if they could all be driven similarly, e.g. common semantics, use of function keys etc, in order that users did not have to do the mental switch and use the help functions."

9A "The interface was a success."

10A "The only real problem with the interface was that it was not complete, this meant greater difficulties in programming for it than were necessary, and the user having to resort to traditional interaction styles too often."

11A "Not all UIs completed since project terminated; for those that were:"

**Good Features:** Simple

Good feedback

Quick.

**Bad Features:** No help

Not for completely naive user.

Demands knowledge of graphics hardware.

12A "Within the limitations of a simple microcomputer, the UI performed well."

**Good Features:** The UI was robust despite the lack of skill of some users; it never got stuck, or failed to respond sensibly to input.

**Bad Features:** The biggest fault was a certain amount of inconsistency between
modules in the UI; "this is not entirely undesirable, as it provided practice in a variety of skills for the user."

3B "The interface is excellent." It is frequently sought for demo's.

**Good Features:** "The graphics output is impressive."

"The UI provides an illustration of what can be achieved with a modern colour graphics workstation and the right software tools."

**Bad Features:** There are flaws because of some limitations in the development package.

4B The UI is "Reasonably satisfactory (about 7/10)."

**Good Features:** "It conforms closely to the accepted house style."

"It is simple, efficient and fairly modeless."

**Bad Features:** "It has to be compromised because of shortcomings in the underlying system."

"Users unfamiliar with the house style may experience difficulties."

"No on-line help apart from a manual page."

5B Design team is sufficiently well pleased with the outcome to embark on a continuing programme of development of more sophisticated software of this type.

**Good Features:** The UI is fast

It is easy to learn and use (it is used in teaching in a university and commercially in the UK and abroad).

**Bad Features:** Does not make use of more sophisticated features which are
available on workstations. Largely, this is because workstation hardware and software has been developed in parallel with this work.

7B "As always, a second go would be nice - however the client wanted a working system. There was little iteration to incorporate user experience into a design change. The client did not want a mouse/WIMP system - they wanted low cost and quick delivery."

**Good Features:** The UI is simple.

Easy for other programmers to change.

**Bad Features:** The UI uses the Xenix libraries. Subsequent updates to the Xenix system have caused software compatibility problems.

1C

**Good Features:** Gathers all information.

Has cross check on data entry.

Good audit procedures.

Easy to follow procedures.

**Bad Features:** Cross check on data entry not liked by clerks - depends on double entry.

2C

**Good Features:** Works efficiently.

Reliable.

Software has not needed major amendments.
Central to system currently used.

**Bad Features**: Lack of agreement in user department on some of the requirements still persists.

Communications support required.

**4C** Not really satisfied. "UI really centred about the use of the VDU."

**Bad Features**: Poor when implemented.

Improved after maintenance/change, but still not good.

Very limited by state of the art of the old VDUs, e.g. no colour, no highlighting, no menus, no mouse, no graphics, only upper case letters etc (this project took place during 1975-1976)

**5C** "It worked. One is quite limited with what one can do with" the application involved.
Appendix 4

Designer Interview Structure From the Interview Based Investigation of Applied and Commercial Design Practice

Stage 1.

1. Designer describes general tasks supported by the system.

2. Designer describes envisaged user population.

3. Designer describes own role in the design process with respect to impact on end-user interface.

Interviewer relies on checklist of points and questions to ensure appropriate coverage.

Stage 2.

General and specific points concerning design decisions discussed. Interviewer to include particular interface characteristics of:

i) The primary system ii) Sources of information used in design iii) Constraints on design activity

Designer to determine the content. Interviewer uses the checklist of general and specific points relating to the system under discussion. If the system in question is available, it should be referred to.
Stage 3.

Designer discusses design philosophy and issues more generally.
Checklist for Interview

User Population

Predictability of user population
Homogeneity/heterogeneity
Age group(s)
Professional or private users
General experience
Previous system training
Planned training
Probable frequency of interactions
Probable Length of interaction sessions

Application(s)

Programming
Design
Drawing
Office tasks
Editing
Teaching
Monitoring
Data analysis
Information retrieval
Project planning
Expert system
Games
Other

System Information presentation

Text lay-out
Windows
Menus
Icons
Graphical
Other

Input Devices

Keypad
Special key functions
Mouse
Joystick
Touch screen
Light pen
Other

Input Methods

Command language
Menu selection
Special tools

User's System Model

Manuals
Help facilities (active/passive)
Error messages
Well-formed metaphor
Consistency/predictability
Modelessness
Discoverability of functionality
Applicability of HCI Techniques to Systems Interface Design

Appendix 5

HCI Practitioner Interview Structure

Stage 1.

1. Interviewee describes own background with respect to HCI and computer system design experience
2. Interviewee describes organisation(s) within which s/he practices/d
3. Interviewee describes Team structure (plus roles) within which s/he works/d

Stage 2.

1. Interviewee lists as many as possible HCI or SAD Techniques applied, successfully or otherwise
2. Interviewee ranks these in order of their usefulness (perceived or otherwise)
3. Starting with the highest ranking, the interviewee discusses each technique in turn.

Stage 3.
For each technique interviewee describes situations within which it was applied in terms of the following:

1. End product function(s), type of interface (e.g. WYSIWYG) & prospective users

2. Design-team familiarity with system application domain

3. Roles of organisations involved (own, client, marketers etc)

4. Your particular role in the design of the interface

5. Human/technological resources (research, hardware, etc) & time resources (e.g. person-hours, deadlines)

6. Collection of information about client requirements, user requirements, tasks, objects, processes etc.

7. Analysis of information

8. Was the technique used in a manner that was independent of or integrated with the rest of the design process

9. Top-down, bottom-up, modular or incremental in approach (which best)

10. The scope of the technique used (aims, targets, output etc)

11. Usability metrics associated with technique (if any)

12. Utility, translatability, compatibility with other specs

13. Comprehensibility by expert, non-expert, user

14. Generation and V&V (formal, informal, iterative, etc). Evaluation (self -, user -, experimental etc)
15. Notable problems & modifications

16. Satisfactoriness of the product when finalised (why finalised)

17. What you would have done differently with the benefit of hindsight

18. Any additional information

Possible constraints on design activities

Interviewee ranks any experienced in descending order of importance (i.e. 1 is most important).

1.___________

2.___________

3.___________

4.___________

5.___________

6.___________

Information Sources

What information sources were used during the design process?

How were these exploited?
Stage 4.

Practitioner discusses design philosophy and issues more generally.

1. What types of techniques and methodologies are generally the most useful and why.

2. Which are the easiest for the HCI Expert to apply.

3. Which are the easiest for the non-expert to apply.

4. Which techniques and methodologies can provide the most comprehensible output for implementors and user-evaluators.

5. Do the properties of a user-tested final prototype form the basis for requirements specification.

6. Do methodologies provide clear information on how to obtain the relevant information required to support their application.

7. What kinds of abstraction/specification are most easily used for representation/evaluation by other members of the design team/clients/users etc.

8. Is there such a thing as a useful evaluative grammar for the user-interface. If so what are the appropriate targets of description, metrics etc.

9. Are tools such as UIMS, Applications Generators etc useful aids.

10. Are tools which model user cognitive processes useful - say how

11. Are HCI design methodologies useful - how

12. Is an approach which can be incorporated with the rest of
design better that one which is independent but which also yields compatible results.