Integrating Human Factors with Structured Analysis and Design Methods

by

Kee Yong Lim

University College London

Submitted for the Degree of Doctor of Philosophy
at the University of London

June 1992

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Abstract

Current human factors input to system development is effected through methods, tools and guidelines. Although the input prompts the consideration of human factors concerns during system design, reports have highlighted inadequacies with respect to the scope, granularity, format and timing of the contributions, e.g. Smith, 1986; Chapanis and Burdurka, 1990; Sutcliffe, 1989; etc.

The thesis argues that such problems are obviated if design needs of both Software Engineering and Human Factors are appropriately represented within an overall system design cycle. Intersecting concerns may then be identified for explicit accommodation by the design agenda. To derive an overall design cycle, current conceptions for the individual disciplines should be examined. Since these conceptions are expressed at a lower level as methods, an overall design cycle may be instantiated more specifically by integrating compatible methods from the two disciplines. Methodological integration is desirable as design inter-dependencies and roles may be defined explicitly. More effective inter-disciplinary communication may also accrue from the use of a common set of notations.

Methodological integration is facilitated if the design scope, process and notation of individual methods are well defined. Such characteristics are found in a class of Software Engineering methods commonly referred to as structured analysis and design methods. Unfortunately, the same are not currently to be found for human factors since its methods are generally unstructured and focus only on later design stages.[1] Thus, a pre-requisite for integration is the derivation of a reasonably complete and structured human factors method. Since well developed Software Engineering methods already exist, it would be appropriate (for the purposes of methodological integration) to structure human factors methods around specific structured analysis and design methods. The undertaking is exemplified by the present research for the Jackson System Development method. In other words, the scope of the thesis comprises the derivation, test and integration of a structured

[1] The imbalance derives from the historically late recruitment of human factors to system design.

human factors method with the Jackson System Development method.

In conclusion, the research contributes to the Human Factors discipline in two respects. Firstly, it informs the research community on how similar work with other structured analysis and design methods may be set up. Secondly, it offers designers an extended Jackson System Development method that facilitates the incorporation of human factors during system development.

CONTENTS

Abstract.....1

Contents.....3

List of Figures.....6

List of Tables.....12

Acknowledgements.....15

Preface.....17

PART I : Research Background.....21

Chapter One : Introduction.....23

Chapter Two : On Human Factors Input during System Development...33

PART II : On Human Factors Integration with Structured Analysis and Design Methods (SADMs).....59

Chapter Three : General Research Requirements, Activities and Plan for Integrating Human Factors with SADMs.....61

Chapter Four : Towards a Conception of Structured Human Factors Design.....78

Chapter Five : A Review of Previous Research into the Integration of Human Factors with SADMs.....105

PART III : On Human Factors Integration with the Jackson System Development Method.....130

Chapter Six : General Research Concerns for Integrating Human Factors with the Jackson System Development Method.....132

Chapter Seven : Research Highlights in Integrating Human Factors with the Jackson System Development Method.....155

PART IV : A Structured Human Factors Method for the Jackson System Development Method.....186

Chapter Eight : An Overview of the JSD* and JSD*(HF) Methods.....188

Chapter Nine : The Elicitation and Analysis Phase of the JSD*(HF) Method.....198

Chapter Ten : The Synthesis Phase of the JSD*(HF) Method.....239

Chapter Eleven : The Design Specification Phase of the JSD*(HF) Method.....273

PART V : Synopsis of the Research.....307

Chapter Twelve : An Assessment of the Present Work and Opportunities for Follow-up Research.....309

PART VI : References.....338

PART VII : Appendices.....356

Glossary.....358
Index of Abbreviations.....361
Annexes.....364
Bibliography.....413

List of Figures

Figure in Preface :

Figure P-1 : Research Concerns and Thesis Structure.....20

Figures in Chapter One :

Figure 1-1 : Cost of Fixing Design Errors Relative to the System Design Cycle.....26

Figure 1-2 : Effort Ratios for Human and Machine Design Relative to the System Design Cycle.....28

Figure 1-3 : Relating the Impact of Human Factors to Various Phases of the System Design Cycle.....29

Figure 1-4 : Relating Design Outcomes to the Duration of Human Factors Involvement at Various Stages of the Design Cycle.....29

Figures in Chapter Two :

Figure 2-1 : Basic Characteristics of Human Factors Design Toolsets.....34

Figure 2-2 : Merits of Rapid Prototyping.....44

Figure 2-3 : Demerits of Rapid Prototyping.....45

Figure 2-4 : Merits of Structured Analysis and Design Methods.....48

Figure 2-5 : Demerits of Structured Analysis and Design Methods.....48

Figure in Chapter Three :

Figure 3-1 : A Research Plan for Specifying an Integrated Method.....72

Figures in Chapter Four :

Figure 4-1 : An Initial Conception of Structured Human Factors Design.....98

Figure 4-2 : A Conception of the Structured Design of Human-Machine Systems (Jones, 1973).....99

Figure 4-3 : An Updated Conception of the Structured Design of Human-Machine Systems (Shackel, 1986a).....101

Figure 4-4 : A Modified Version of Shackel's (1986a)

Conception of Structured System Design.....103
Figure 4-5 : An Enhanced Conception of Structured
Human Factors Design.....104

Figures in Chapter Five :

Figure 5-1 : Schematic Representation of a Human Factors Method
that Complements the SASD Method (Blyth and Hakiel, 1988).....108
Figure 5-2 : Function Allocation using a System Function Model.....111
Figure 5-3 : Schematic Representation of a Human Factors
Method that Complements DIADEM (Damodaran et al, 1988).....116
Figure 5-4 : Checklists, Questionnaires, Observation Sheets
and Summary Forms for User Analysis in DIADEM.....117
Figure 5-5 : Design Steps of the Task Charting Technique.....118
Figure 5-6 : DIADEM's Task Allocation Chart Notation.....119
Figure 5-7 : JSD Initial Model Extended to Include a
Human-Computer Interaction Layer (Carver et al, 1987).....126
Figure 5-8 : Screen Layout Description Corresponding to
the Extended JSD Initial Model in Figure 5-7.....127

Figures in Chapter Six :

Figure 6-1 : Basic Constructs of JSD Structured Diagram Notation.....135
Figure 6-2 : A Schematic Representation of JSD
Specifications (Renold, 1989).....139
Figure 6-3 : An Instantiation of the General Research Plan
for Human Factors Integration with the JSD Method.....148

Figures in Chapter Seven :

Figure 7-1 : A Summary of Research Activities for Human
Factors Integration with the JSD Method.....156
Figure 7-2 : Using JSD Structured Diagrams to Link Screen Object
Behaviours to Corresponding Changes in their Appearance.....162
Figure 7-3 : Initial JSD* and JSD*(HF) Conceptions.....166
Figure 7-4 : A Product-Oriented Conception of JSD*.....167
Figure 7-5 : Scope of the Review of Current Concerns of

Human Factors Design.....168
Figure 7-6 : Setting Human Factors Models Against the System Design Life-Cycle.....175
Figure 7-7a : Further JSD* Conceptions I.....178
Figure 7-7b : Further JSD* Conceptions II.....179
Figure 7-8 : 'Basic Elements' for Describing a User Interface Design.....180
Figure 7-9a : Early Versions of the JSD* Method I.....182
Figure 7-9b : Early Versions of the JSD* Method II.....184

Figures in Chapter Eight :

Figure 8-1 : Locating the JSD*(HF) Method within the JSD* Method.....189
Figure 8-2 : Block Diagram Summary of Each Design Stage of the JSD*(HF) Method.....196

Figures in Chapter Nine :

Figure 9-1 : Block Diagram Summary of the Extant Systems System Analysis (ESSA) Stage.....201
Figure 9-2 : Extant System Categories Assumed by the JSD*(HF) Method.....206
Figure 9-3 : Task Description for Network Security Management at University College London Computer Centre (TD(UCLCC)) -- Pages 1-4.....213
Figure 9-4 : Task Description for PC Security Management for the MacPassword™ Application (TD(MPASS)) -- Pages 1-2.....220
Figure 9-5 : Generalised Task Model Description for Network Security Management at University College London Computer Centre (GTM(UCLCC)) -- Pages 1-3.....226
Figure 9-6 : Block Diagram Summary of the Generalised Task Model (GTM) Stage.....230
Figure 9-7 : Generalised Task Model of Network Security Management for the Target System.....233
Figure 9-8 : Generalised Task Model Description for an Extant System Composite (GTM(x)) -- Pages 1-3.....236

Figures in Chapter Ten :

Figure 10-1 : Block Diagram Summary of the Statement of User Needs (SUN) Stage.....241
Figure 10-2 : Part of SUN(y) for Network Security Management.....243
Figure 10-3 : DoDD(y) for Network Security Management.....246
Figure 10-4 : Block Diagram Summary of the Composite Task Model (CTM) Stage.....249
Figure 10-5 : CTM(y) for Network Security Management -- Pages 1-3.....252
Figure 10-6 : Design Products Exchanged between JSD*(SE) and JSD*(HF) Methods at the First Inter-Dependency Point.....260
Figure 10-7 : Block Diagram Summary of the System and User Task Model (SUTaM) Stage.....265
Figure 10-8 : Part of STM(y) for Network Security Management -- Pages 1-3.....267
Figure 10-9 : Design Products Exchanged between JSD*(SE) and JSD*(HF) Methods at the Second Inter-Dependency Point.....271

Figures in Chapter Eleven :

Figure 11-1 : Block Diagram Summary of the Interaction Task Model (ITM) Stage.....275
Figure 11-2 : Part of ITM(y) for Network Security Management -- Pages 1-2.....278
Figure 11-3 : Pictorial Screen Layout of Screen 3A.....280
Figure 11-4 : Pictorial Screen Layout of Screen 3B.....281
Figure 11-5 : Dimensioned Pictorial Screen Layout of Screen 3A.....286
Figure 11-6 : Block Diagram Summary of the Interface Model (IM) and Display Design (DD) Stages.....288
Figure 11-7 : Part of DITaSAD(y) for Network Security Management.....293
Figure 11-8 : Pictorial Screen Layout of Screen 4B.....294
Figure 11-9 : IM(y) Description of Radio Buttons in Screen 4B.....295
Figure 11-10 : Pictorial Screen Layout of Screen E1.....296
Figure 11-11 : Pictorial Screen Layout of Screen 5B-1.....299
Figure 11-12 : IM(y) Description of Item(s) in the 'User Name Display' Window -- Screen 5B-1.....300

Figure 11-13 : IM(y) Description of a 'Generic' Radio Button in the 'User List' Window -- Screen 5B-1.....301
Figure 11-14 : IM(y) Description of the 'Show' Button -- Screen 5B-1.....302
Figure 11-15 : Pictorial Screen Layout of Screen 5B-2.....303
Figure 11-16 : Pictorial Screen Layout of Screen C2.....305

Figure in Chapter Twelve :

Figure 12-1 : Locating the JSD*(HF) Method within a Range of Human Factors Support for System Development.....331

Figures in Annex A :

Figure AA-1 : Constructs of JSD and JSD* Structured Diagram Notation.....365
Figure AA-2 : JSD and JSD* Structured Diagram Descriptions of a Fictitious Task 'T' 365

Figure in Annex B :

Figure AB-1 : Scenarios for Applying Generification.....367

Figures in Annex C :

Figure AC-1 : The Hierarchical Conception of Work Assumed by the JSD*(HF) Method.....371
Figure AC-2 : Task Classes and their Relationship to Human and Computer Components of a System.....373

Figures in Annex D :

Figure AD-1 : Extract from the Extant Statement of User Needs for the University College London Computer Centre (SUN(UCLCC)).....379
Figure AD-2 : Extract from the Extant Statement of User Needs for the MacPassword™ Application (SUN(MPASS)).....380
Figure AD-3 : Domain of Design Discourse Description for the University College London Computer Centre (DoDD(UCLCC)).....381
Figure AD-4 : Domain of Design Discourse Description for the MacPassword™ Application (DoDD(MPASS)).....383

Figures in Annex E :

Figure AE-1 : Extant Task Description for the Recreation Facility Booking System....388
Figure AE-2 : Domain of Design Discourse Description for the Target Recreation Facility Booking System (DoDD(y))....393
Figure AE-3 : Composite Task Model for the Target Recreation Facility Booking System (CTM(y))....395
Figure AE-4 : Design Inter-dependencies between Software Engineering and Human Factors Streams of the JSD* Method....396
Figure AE-5 : Decomposition of CTM(y) into STM(y) and UTM(y)....399
Figure AE-6 : System Task Model for the Target Recreation Facility Booking System (STM(y))....400
Figure AE-7 : Deriving ITM(y) by Decomposing Human (H:) Leaves of STM(y) on the Basis of the Domain of Design Discourse Description and the Chosen User Interface Environment....404
Figure AE-8 : Interaction Task Model for the Target Recreation Facility Booking System (ITM(y))....405
Figure AE-9 : Deriving IM(y) by Decomposing Computer (C:) Leaves of STM(y) on the Basis of the Domain of Design Discourse Description and the Chosen User Interface Environment....407
Figure AE-10 : Inter-Linkages between Products of the Display Design Stage....408
Figure AE-11 : Dialogue and Inter-Task Screen Actuation Description for the Target Recreation Facility Booking System (DITaSAD(y))....409
Figure AE-12 : A Pictorial Screen Layout Description for the Target Recreation Facility Booking System....410

List of Tables

Tables in Chapter Four :

Table 4-1 : Reported Human Factors Concerns and their Approximate Sequencing during System Design I.....82

Table 4-2 : Reported Human Factors Concerns and their Approximate Sequencing during System Design II.....83

Table 4-3 : Reported Human Factors Concerns and their Approximate Sequencing during System Design III.....84

Table 4-4 : Reported Human Factors Concerns and their Approximate Sequencing during System Design IV.....85

Table 4-5 : A 'Basic' Set of Human Factors Concerns Identified from Reports Surveyed in Tables 4-1 to 4-4.....86

Table 4-6 : Deriving the Relative Sequence of Human Factors Design.....87

Table in Chapter Six :

Table 6-1 : Human Factors Deficiencies of Existing Structured Analysis and Design Methods (after Anderson, 1988).....141

Tables in Chapter Seven :

Table 7-1 : Profile of Case-Study Applications Used in the Research.....157

Table 7-2 : Profile of Research Developments and Reports.....158

Table 7-3 : Information Description Capability of Existing JSD Notations...159

Table 7-4 : Scope of Description of Various Models of Human-Computer Dialogue (after Hartson and Hix, 1988).....171

Table 7-5 : Comparison of User Interface Design Concerns of CLG (Moran, 1981) and the JSD*(HF) Method.....171

Table 7-6 : Comparison of User Interface Design Concerns of Foley and van Dam's Method (1982) and the JSD*(HF) Method.....172

Table 7-7 : Comparison of Newman's (1988) Conception of User Interface Design 'Practice' and the JSD*(HF) Method.....177

Tables in Chapter Nine :

Table 9-1 : TD(UCLCC) Table -- Pages 1-3.....217
Table 9-2 : TD(MPASS) Table -- Pages 1-2.....222
Table 9-3 : GTM(y) Table.....234

Tables in Chapter Ten :

Table 10-1 : DoDD(y) Table.....246
Table 10-2 : CTM(y) Table.....255
Table 10-3 : Event Table for Network Security Management.....257
Table 10-4 : Part of a Functions List for Network Security Management.....261
Table 10-5 : STM(y) Table.....270

Tables in Chapter Eleven :

Table 11-1 : Objects Dictionary -- Screen 4B.....295
Table 11-2 : Objects Dictionary -- Screen E1.....296
Table 11-3 : Part of DET(y) for the Target Network
Security Management System.....296
Table 11-4 : Objects Dictionary -- Screen 5B-1.....300
Table 11-5 : Objects Dictionary -- Screen 5B-2 (Pages 1-2).....304
Table 11-6 : Objects Dictionary -- Screen C2.....306

Table in Chapter Twelve :

Table 12-1 : Comparing JSD* Against Other Integrated Methods.....323

Tables in Annex D :

Table AD-1 : DoDD(UCLCC) Table -- Pages 1-2.....382
Table AD-2 : DoDD(MPASS) Table.....384

Tables in Annex E :

Table AE-1 : Extant Task Description Table for the Recreation
Facility Booking System.....389
Table AE-2 : Dictionary of Screen Objects for the Target
Recreation Facility Booking System (Screen 1).....411

Table in Bibliography :

Table B-1 : Authorship Summary for RARDE Project

Working Documents.....419

Acknowledgements

I am indebted primarily to Professor John Long for his exertions as my supervisor, and for his limitless enthusiasm and support (both moral and intellectual) for this work.

My gratitude is extended to Nigel Silcock for his contributions to parts of this work; and in particular to his bravery as the first 'subject' (perhaps unknowingly) in 'informal tests' on the learnability and utility of the structured method proposed in this thesis.

I am also grateful to Andy Whitefield and Andrew Life for their encouragement and effort in reviewing the structured method in its final form.

Thanks are also due to all members of the Ergonomics Unit for creating such a congenial and pleasant work environment.

I am happy to acknowledge the indirect financial support provided by the Procurement Executive, Ministry of Defence (Royal Armaments Research and Development Establishment, RARDE).[2] Special acknowledgements are due to individual members of staff at CA2 (RARDE)[3] for their contributions in priming the research.

Last but not least, I wish to thank close friends whose encouragement and company, made life during this period a balanced and enriching experience.

[2] It should be noted that views expressed in the thesis are those of the author and should not be attributed to the Ministry of Defence.

[3] Military regulations do not permit citing their names.

To my family, in particular to my mother, whose endless love, patience and support allowed the fruition of this work.

Preface

"We (Ergonomists) borrow and invent techniques to serve our special needs."

A. Chapanis, 1990.

To achieve the objective of developing a more effective means for human factors input to system design, existing problems and solutions should be examined. Such an examination would :

(a) highlight current problems with respect to the role of human factors in system design, i.e. *'who', 'what', 'when'* and *'how'* issues;
(b) support the formulation of promising solutions to observed problems;
(c) indicate how the specification of a solution may be facilitated.

The thesis proposes that one solution for improving human factors contributions *throughout* the design cycle is to locate its inputs against the explicit design framework of Software Engineering structured analysis and design methods. In addition, similarly structured human factors methods could be developed for integration with particular structured analysis and design methods. Complementary Software Engineering and Human Factors design roles are thus defined explicitly with respect to the stage-wise scope, process and notation of the integrated method.

To this end, the thesis is divided into five parts which address the following concerns :

Part I : Research Background -- describes problems of existing approaches for human factors input to system development. The potential of structured analysis and design methods for supporting the development of a solution is then described. The research scope is thus defined as the development and subsequent integration of a structured human factors method with a particular Software Engineering structured analysis and design method.

Part II : On Human Factors Integration with Structured Analysis and Design Methods (SADMs) -- describes general requirements and the research entailed by methodological integration. Specifically, a general research plan is proposed to exemplify how such research could be conducted. Lower level activities of the plan are then described, e.g. a survey of previous reports on human factors design conceptions and human factors integration with structured analysis and design methods.

Part III : On Human Factors Integration with the Jackson System Development Method -- two chapters in this part recount how general requirements and considerations set out in Part II are instantiated and realised during the integration of human factors with the Jackson System Development method. For instance, characteristics of the latter method are examined to identify its requirements for human factors support. Constraints specific to this research are then introduced to condition the *general* research plan proposed in Part II. A *specific* plan is thus derived. Research milestones corresponding to the specific plan are then reviewed.

Part IV : A Structured Human Factors Method for the Jackson System Development Method -- the main product of the research, i.e. the extended Jackson System Development (or JSD*) is presented. As the Software Engineering component of the integrated method is largely unchanged (i.e. the Jackson System Development method), the thesis focuses primarily on the human factors component of the method (namely, the structured human factors method). However, obligatory design inter-dependencies between the two components of the integrated method are highlighted.

Part V : Synopsis of the Research -- three concerns characterise this part of the thesis, namely how the present work should be assessed; the outcome of these assessments; and what follow-up research may be undertaken.

Part VI : References -- this part provides an alphabetical listing of all references cited in the thesis.

Part VII : Appendices -- the appendices support the main text of the thesis. A list of publications and internal reports of the research is also provided.

The above structure of the thesis is summarised graphically in Figure P-1.

In conclusion, the present work contributes to the Human Factors discipline in two respects :

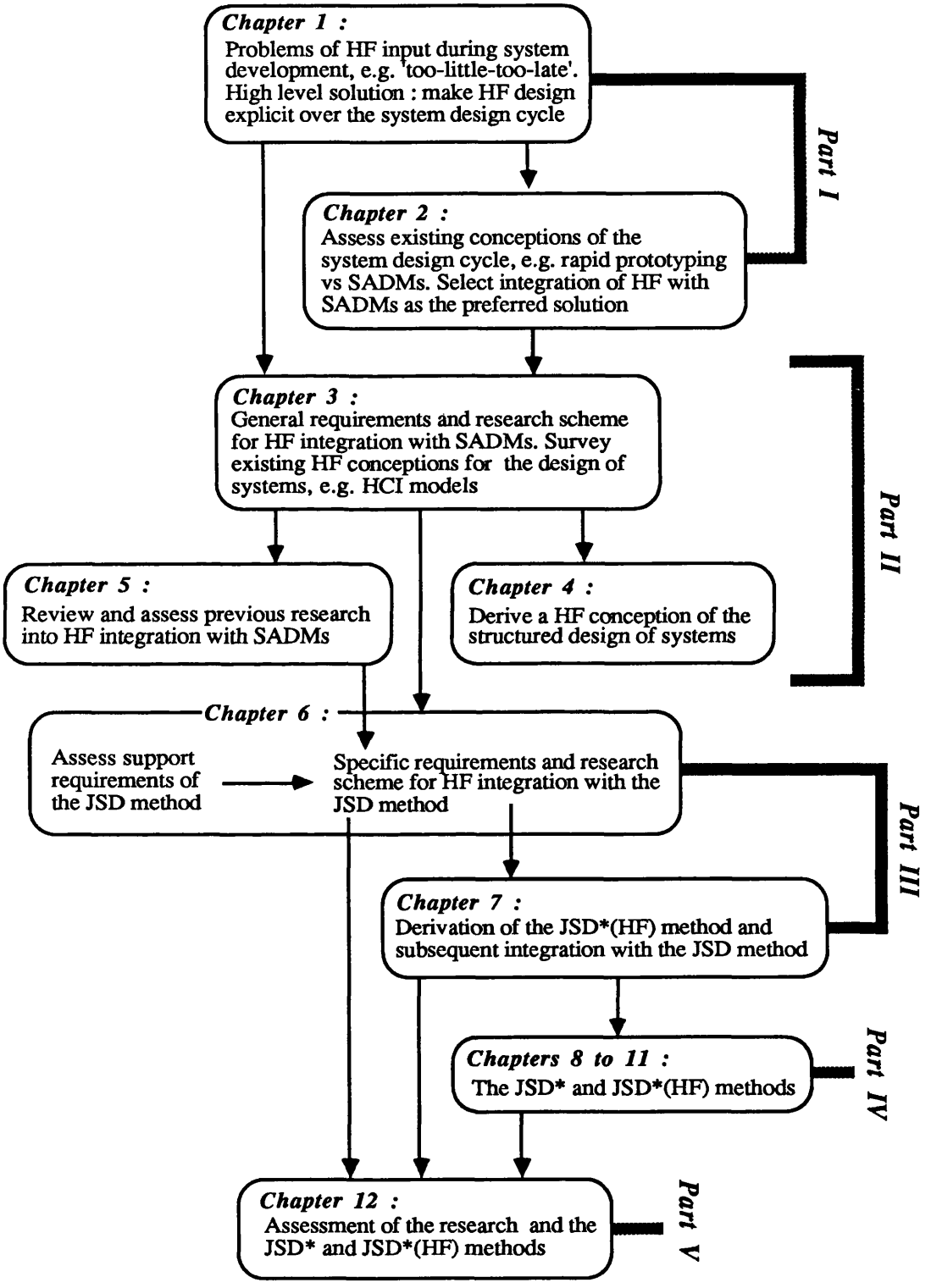
(a) it informs the *academic* community on how similar research may be conducted;
(b) it offers *human factors designers* an extended Jackson System Development method for system design.

These contributions are organised in the thesis as follows :

(1) for group (a) above, Parts I, II and V of the thesis would be directly relevant. Part III may also be informative.
(2) for group (b) above, Part IV provides a detailed account of the extended Jackson System Development method.

Kee Yong Lim,
June 1992.

Figure P-1 : Research Concerns and Thesis Structure



PART I :

Research Background

CONTENTS

Chapter One : Introduction.....23

1.1. Current Problems of Human Factors Input to Systems Design.....23
1.2. Preliminary Scope and Objectives of the Research.....32

Chapter Two : On Human Factors Input during System Development.....33

2.1. An Assessment of Current Solutions for Supporting Human Factors Input into System Development.....34
2.2. Conceptions of the System Design Cycle : Traditional System Development Life-Cycle versus Rapid Prototyping 42
2.3. A Case for Integrating Human Factors with Structured Analysis and Design Methods.....51
2.4. Detailing the Scope and Objectives of the Research.....57

Chapter One : Introduction

".....in the design domain we can never know enough."

Rosson, 1985.

"If at first you know where you are, and whither you are tending, you will better know what to do and how to do it."

Abraham Lincoln, 1809-1865.

This chapter reviews current problems of human factors input to system design. The research context is thus characterised. For instance, the review indicated that current human factors contributions are poorly timed relative to design support requirements that vary throughout the design cycle. Thus, the relevance, format and granularity of its inputs are non-optimal for effective uptake of human factors inputs. Promising solutions may then be proposed to address the observed problems.

In other words, Chapter One sets the context for Chapters Two to Five which considers *what* improvements to existing means of human factors input are necessary, and *how* such improvements could be achieved.

1.1. Current Problems of Human Factors Input to Systems Design

Recent developments in computer technology has resulted in a shift from mainframes to interactive personal computers, e.g. the availability and affordability of personal computers and the rapid diversification in computer applications. Today, personal computers have made significant inroads into both the workplace and the home. Thus, the computer user base is widened considerably. The wider user base, together with market forces, highlighted the importance of designing computer applications that are appropriate in both functionality and usability. The success of Macintosh computers is an example (see also Shackel, 1985 and 1986b; CCTA (Draft) Report, 1988, Annex 1; Shuttleworth, 1987). In addition, it was

recognised that poor usability implies :

(a) greater training requirements which is a particularly serious problem with a highly mobile information technology work force;
(b) an incomplete utilisation of the software (Hirschheim, 1985; Stevens, 1983);
(c) possible failure of the system to achieve its intended purpose (Underwood, 1987; Lucas, 1975);
(d) slower than anticipated uptake of information technology in the workplace (Fáhnrich et al, 1984; Eason and Cullen, 1988).

In short, poorly designed systems do not compete well in the market. The importance of user testing, and hence human factors, was thus recognised. This form of human factors recruitment was established and its designers were called upon only to *evaluate* the usability of a particular design (Rosson, 1987). Correspondingly, human factors methods gravitated towards later stages of system development. It appears that the need for wider human factors involvement was unrecognised until Software Engineering design processes were made more explicit. Thus, recognition may have been triggered by the following events :

(a) the advent of more powerful computers that led to the development of larger and more complex systems. Such developments constitute longer term projects with multiple design deliverables and a greater propensity to over-run specified deadlines;
(b) the penetration of software applications into novel domains (i.e. domains which are ill defined) as opposed to the direct computerisation of manual systems (Benyon and Skidmore, 1987; Galliers, 1984);
(c) the emergence of expert systems and artificial intelligence.

The greater demands on the system design process imposed by these events highlighted the following requirements :

(a) to ensure that system development is 'correctly' conducted, e.g. the

adoption of a systematic design approach to enforce orderly design development, modification and post-delivery maintenance; continuous verification of design specifications;

(b) to ensure that a valid or 'correct' system is developed, i. e. a more reliable means of capturing and confirming user requirements;

(c) to increase design consistency, throughput and capability, e.g. the development of computer-based tools to reduce system delivery times; the employment of greater manpower to handle larger scale projects (i.e. emphasis is on design teams as opposed to individual hackers);

(d) to ensure better project management.

These design requirements may be mapped onto the following Software Engineering solutions :

(a) design principles and techniques, e.g. delaying design commitment, incremental design, rapid prototyping, software re-usability;

(b) executable specifications and high level languages, e.g. new generation languages (Fourth and Fifth Generation Languages);

(c) formal notations and methods, e.g. Z, Communicating Sequential Processes (CSP), Calculus of Communicating Systems (CSP), Vienna Development Method (VDM);

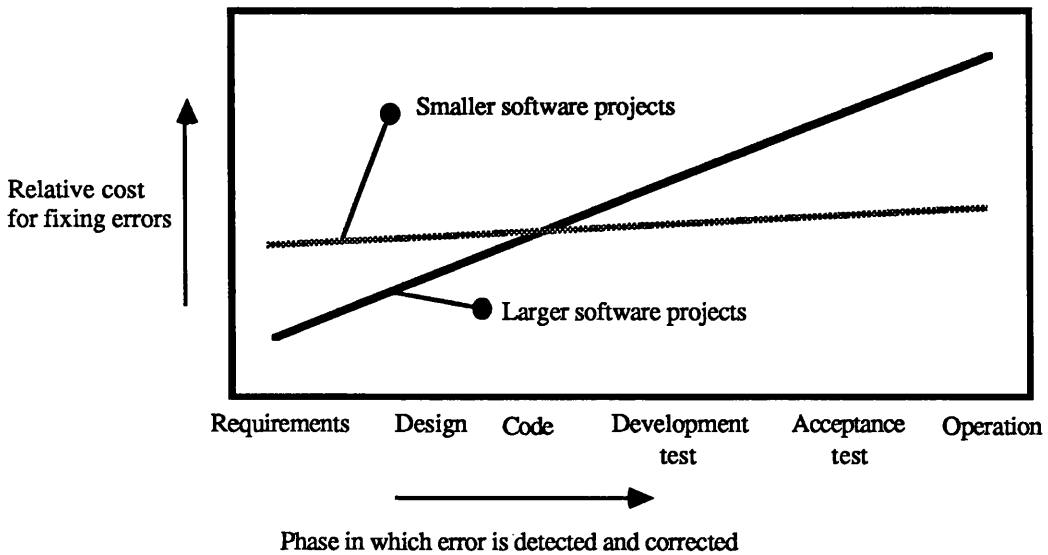
(d) structured analysis and design methods, e.g. Structured Systems Analysis and Design Method (SSADM), Jackson System Development (JSD) method;

(e) computer-based tools, e.g. Computer Aided Software Engineering (CASE), Integrated Project Support Environment (IPSE).

These solutions support software engineers in a number of ways. For instance, more reliable specification through proveability via formal methods; more efficient and effective utilisation of design resources via directly executable notations and computer-based tools (such as code generators and consistency checkers); better design management and more systematic design analysis via structured analysis and design methods; etc.

These Software Engineering developments led to a more explicit definition of the system design process. Thus, it became apparent that human factors involvement only at the evaluation stage was inadequate for ensuring a satisfactory design outcome, e.g. an efficient design of a valid system. Specifically, effective human factors contributions may become inordinately difficult at this late design stage. For instance, the formulation of human factors contributions may be hampered by poorly documented design rationale and decisions. Thus, late human factors involvement may result in contributions which are 'too-little', i.e. the contributions comprise little more than advice. Since late involvement implies that human factors activities do not constitute a basic part of design specification, its contributions may also be 'too-late', i.e. the recommendations could not be acted upon (Rosson, 1987). For instance, actions could be thwarted because desired modifications are too far reaching and hence excessively difficult and expensive to implement (see Figure 1-1). Human factors recommendations may be too far reaching as a result of early errors being magnified through subsequent design stages, i.e. a progressive degradation of the design specification (Alvey MMI Workshop Report, 1984, pg. 20). Alternatively, modifications may be restricted because fully developed designs

Figure 1-1 : Cost of Fixing Design Errors Relative to the System Design Cycle[1]



[1] From Bóhm (1981).

become 'frozen' by inter-locking dependencies and are thus more resistant to change (Grudin et al, 1987; Bury, 1984). Such ineffective design development has resulted in the consumption of a disproportionately large amount of project resources by design maintenance,[2] e.g. up to 70% of project resources may be consumed for this purpose alone (Bóhm, 1976 and 1981; Multi-User Computing, 1989, pg. 20).[3, 4]

Thus, to avoid the 'too-little-too-late' problem human factors involvement should not only be early but sustained throughout the design cycle (see also Alvey Human Interface Committee Report, 1987, pg. 9; Eason and Cullen, 1988). Early design involvement ensures more effective human factors input because human issues which predominate at early design stages (Figure 1-2) may then set the constraints for later design stages which largely concern the device (Alvey MMI Workshop Report, 1984, pg. 20). Continuous involvement ensures correct translation and incorporation of human factors inputs into the design. It also ensures that inputs are timely and contextually relevant to design concerns at each stage of the design cycle. For instance, wider involvement would support the incorporation of human factors contributions during functional design, e.g. contributions may include descriptions of user goals, tasks and abilities, and these constitute an appropriate basis for functional design. Thus, human factors became involved in the specification of system usability and functionality (as opposed to only evaluation). In other words, human factors may contribute to the analysis, specification, implementation and evaluation of systems, i.e. its contributions permeate the entire

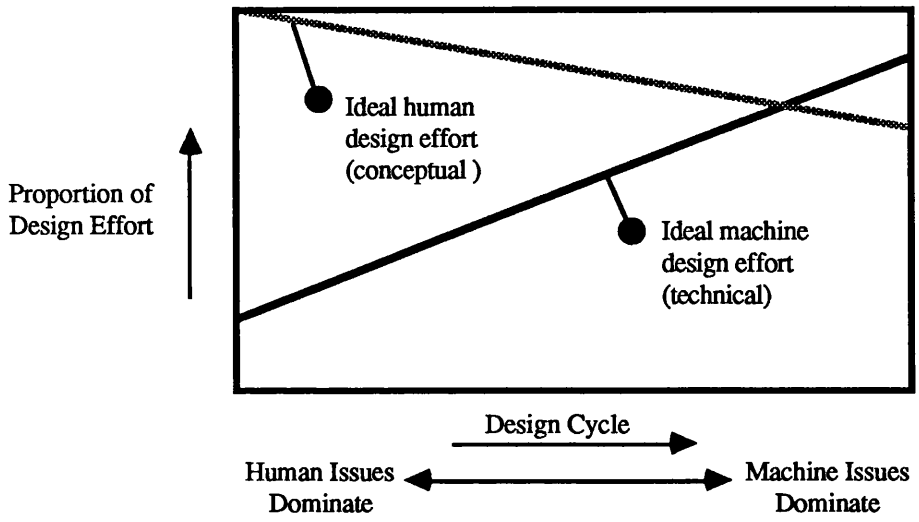
---

[2] Design maintenance, by definition, includes both corrective actions resulting from design errors, and modifications and/or enhancements arising from changes in initial design requirements (Fitzgerald, 1988).

[3] Other accounts report that : large software projects cost, on average, twice as much as their initial budgets; they are usually completed a year late; and a quarter of the projects are never completed at all (Multi-User Computing, 1989, pg. 22).

[4] The contribution of human factors to the reduction of maintenance costs has been reported by Norman (1986), and Rubinstein and Hersh (1984).

Figure 1-2 : Effort Ratios for Human and Machine Design Relative to the System Design Cycle[5]



design cycle (see Alvey Human Interface Committee Report,[6] 1987, pg. 32; Alvey MMI Workshop Report, 1984, pg. 37 & 39; Buxton et al, 1983). The design impact of human factors is thus maximised by earlier and wider design involvement (see Figures 1-3 and 1-4).

Unfortunately, these positive design developments could not be exploited directly due to the following reasons :

(a) to exploit the positive developments, corresponding developments in human factors are required, namely :

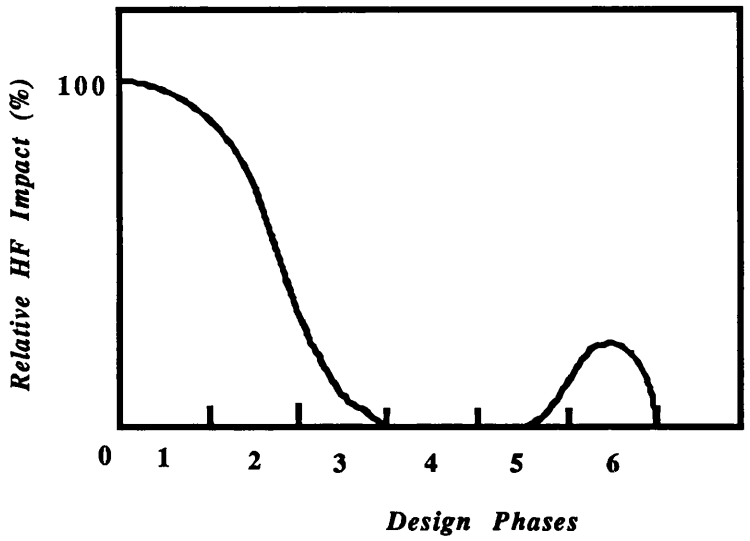
(i) the scope of human factors means of design input (e.g. standards, methods, etc. -- henceforth referred to collectively as 'toolsets' (see Chapter Two)) should be extended to rectify the historically narrow

---

[5] Adapted from Alvey MMI Workshop Report (1984), pg. 20.

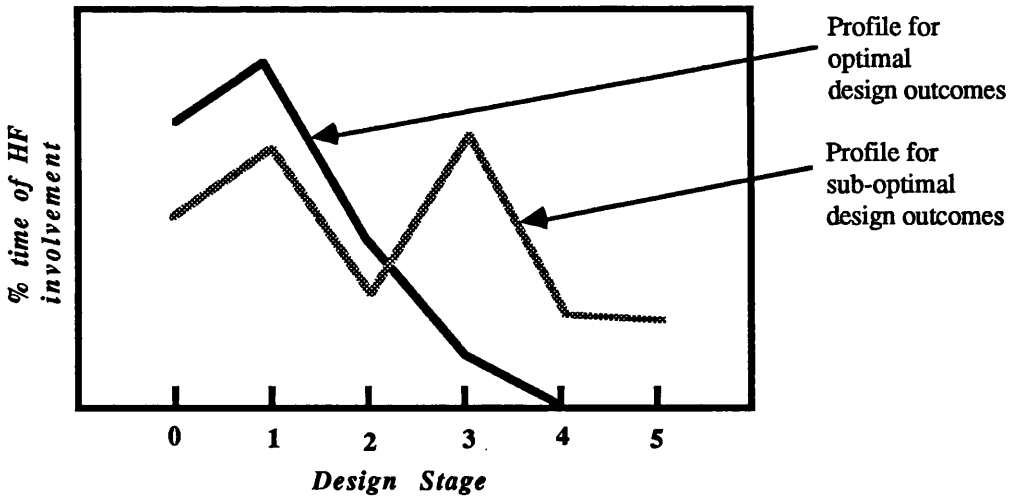
[6] The report emphasised that significant impact on product design will not accrue unless human factors is incorporated in all phases of the design cycle.

Figure 1-3 : Relating the Impact of Human Factors to Various Phases of the System Design Cycle[7]



1 : Conceptual Phase; 2 : Prototype; 3 : Detailed Version; 4 : Production; 5 : Evaluation; 6 : Operation.

Figure 1-4 : Relating Design Outcomes to the Duration of Human Factors Involvement at Various Stages of the Design Cycle[8]



0 : Investigate; 1 : Design; 2 : Prototype; 3 : Implement; 4 : Test; 5 : Release.

---

[7] From Moraal and Kragt (1990).

[8] From Lundell and Notess (1991).

coverage of the system design cycle (ALV/MMI/PRJ/143, 1988, pg. 3). The extension is necessary as human factors is reported to be 'lacking in methodology' (Klein and Newman, 1987);

(ii) human factors design should be made more explicit and complete (i.e. to elevate its practice from craft to engineering status (Long and Dowell, 1989);

(iii) human factors contributions should be more specific to rectify current problems in their translation into design specifications, e.g. current inputs focus on what should be done but not how (Sutcliffe, 1989); the format of inputs do not support effective communication (Mantei and Teorey, 1988).

Since the above developments are extensive, a significant time lag is incurred between the supply and demand of human factors design support (see also Underwood, 1987; CCTA (Draft) Report, 1988, Annex 1; Galliers, 1984);

(b) the exploitation of positive developments was hampered by a number of mismatches between Software Engineering and Human Factors design approaches and activities, e.g. machine-centered versus user-centered design. These mismatches arose because previous solutions for improving system development were formulated independently by the individual disciplines.[9] To facilitate wider human factors design involvement, existing solutions for improving system development should thus be reconciled across the disciplines (Long and Dowell, 1989;[10] Carroll and

[9] Grudin et al (1987) observed that in general, human factors is either inadequately represented or omitted from Software Engineering literature. Human factors has also been 'perceived as irrelevant to system development' (Klein and Newman, 1987).

[10] Human Computer Interaction, as a discipline comprising Human Factors and Software Engineering (Long and Dowell, 1989), is compatible with this perspective.

Campbell, 1986[11]).

The above considerations were summarised aptly by Carroll and Campbell (1986) as follows :

> *"Human-computer interaction.....will favour inter-disciplinary co-operation between psychology and computer science.....will not sustain approaches that are too low level, too limited in scope, too late, and too difficult to apply in real design....."*

Since existing Software Engineering toolsets are better developed, it follows that human factors toolsets should be constructed to augment them (this is also consistent with the current supporting role of human factors). To this end, human factors research should focus on the following :

> (a) in the short term, the development of a means of compensating for the current incompleteness of human factors knowledge, e.g. approximation and empirical techniques for deriving design information that cannot be prescribed analytically;
> (b) definition of a complete human factors design cycle. This definition would make human factors design roles and responsibilities more explicit;
> (c) identification of design relationships between Software Engineering and Human Factors *throughout* the system design cycle. Making such design relationships explicit would facilitate human factors involvement throughout the design cycle;
> (d) development of appropriate toolsets to support the incorporation of human factors inputs *throughout* the system design cycle. The configuration of such toolsets should complement existing Software Engineering design toolsets. Specifically, existing design notations, techniques, methods and tools should be taken into account during the

---

11 As per Footnote 10, but Carroll and Campbell's (1986) conception of Human Computer Interaction comprises Computer Science and Psychology. The difference may be attributed to the scope assumed for the component disciplines.

construction of human factors toolsets; [12]

(e) in the long term, the accumulation of further human factors design knowledge.[13]

The above research agenda has also been proposed elsewhere, e.g. Alvey Human Interface Committee Report (1987); Alvey MMI Workshop Report (1984); Glasson (1984); Mantei and Teorey (1988); Grudin, Ehrlich and Shriner (1987); Klein and Newman (1987); Robertson (1987); CCTA (Draft) Report (1988, Annex 1); McKeen (1983); Innocent (1982); etc.

Preliminary objectives for the present work were then defined with respect to the research agenda.

1.2. Preliminary Scope and Objectives of the Research

Preliminary objectives of the research comprise the following :

(a) to assess the scope and configuration of current human factors toolsets for supporting system development, and to identify problems reported in their use. The objective of the assessments is to characterise the problems that are addressed by the research (see Chapter Two);

(b) to propose and demonstrate, on the basis of (a) above, methodological integration as a means of improving human factors input to system design. The demonstration comprises several design case-studies involving the development and application of a structured human factors method. The latter method is constructed specifically for integration with a particular Software Engineering method (see Part II).

A detailed account of the research entailed by these objectives follows.

---

[12] Such a perspective is consistent with existing configurations of design teams, i.e. system design is led by software engineers with human factors designers providing collaborative support.

[13] item (e) does not fall within the scope of the research. It is included here for completeness.

Chapter Two : On Human Factors Input during System Development

"All things exist in time. They are not unchanging, and they cannot be designed without regard for the way they operate and are used over time."
Charles Owen, 1986, ***Design Processes Newsletter.***

"To every Form of being is assigned', Thus calmly spoke the venerable Sage, 'An active principle.'"
William Wordsworth, 1814, ***The Excursion.***

Having characterised the research scope, existing human factors toolsets should be assessed to determine *what* enhancements would ensure a more effective input to system design. Three complementary classes of human factors toolsets will be assessed briefly, namely :

(a) principles, guidelines, and standards;
(b) computer-based tools;
(c) techniques and methods.

Following the review, greater emphasis will be placed on (c) as a member of its class, namely Structured Analysis and Design Methods (SADMs), could provide frameworks to support the development and recruitment of (a) and (b) above. Specifically, design support requirements at various stages of the design cycle are defined explicitly by the scope, process and representation of SADMs. On this basis, HF toolsets could be contextualised appropriately to system design needs. Similarly, toolsets could be organised more effectively to maximise their symbiosis. Thus, the weaknesses of one are compensated appropriately by the other, while strengths are jointly exploited, e.g. complementary roles for procedural and declarative toolsets may be defined explicitly at various stages of the design cycle. Consequently, a case is made for integrating human factors methods with SADMs. The preliminary scope and objectives of the research are then expanded to support the proposed integration of Human Factors and Software Engineering methods. A more detailed account of *how* methodological integration may be achieved is

described in Chapters Three to Five.

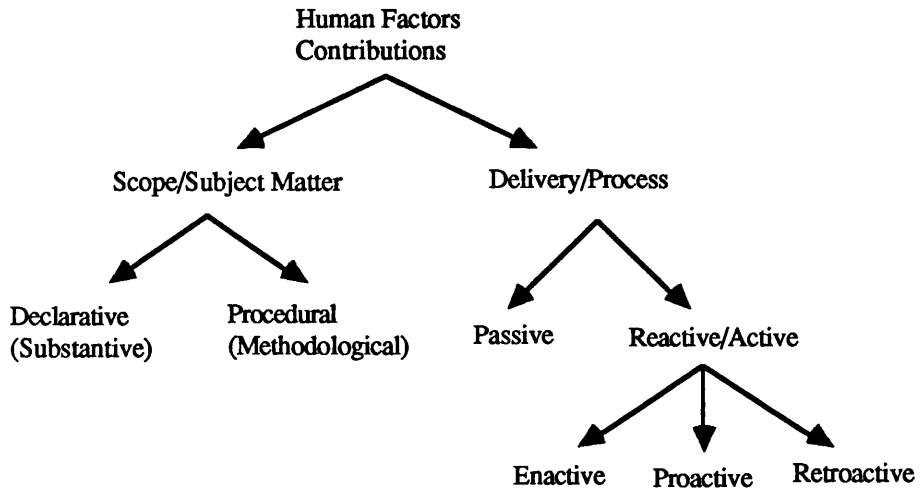
In summary, the objectives of the chapter comprise the following :

(a) to describe and contrast existing classes of human factors toolsets;
(b) to identify typical problems of the classes of toolsets in (a) above;
(c) to highlight required enhancements of the classes of toolsets so that the problems identified in (b) above could be obviated;
(d) to propose and assess methodological integration as a solution with respect to (b) and (c) above.

2.1. An Assessment of Current Solutions for Supporting Human Factors Input into System Development

Existing human factors toolsets may be characterised in terms of their scope (subject matter) and process (delivery) (Figure 2-1).

Figure 2-1 : Basic Characteristics of Human Factors Design Toolsets



A brief account of these characteristics follows :

(a) scope (subject matter) -- declarative-procedural dimension. If the subject matter of a toolset is concerned directly with attributes of the artefact to be designed, its support is termed declarative (or substantive). General, requirements of declarative toolsets comprise the provision of the information that is :

(1) correct and relevant to the design concerns at hand;
(2) presented using appropriate notational schemes so that the information is in an easily accessible and concise format;
(3) expressed at the appropriate level of description.

A toolset may alternatively provide procedural (or methodological) guidance on how a design of the artefact should be derived. For instance, a set of intermediate design representations and transformations may be recommended to support problem analysis and design development. General requirements of procedural toolsets are that its design representations and transformations should :

(1) be compatible with established design practices;
(2) cover the design cycle adequately;
(3) be expressed appropriately to support design reasoning and the management of design complexity. Too low a level of expression of the toolset may impose undesirable constraints on designers (e.g. methods have been criticised for hampering designer creativity), and limit its applicability across different design scenarios. Conversely, too high a level of expression of the toolset may result in inadequate design guidance, e.g. some methods are no more than checklists.

Since procedural support is not a substitute for human factors knowledge, declarative support is invariably required in some form.

(b) process (delivery) -- passive-active dimension. This dimension describes how and when human factors contributions may be delivered during system development, e.g. handbooks are passive in their delivery, while computers and consultants are active or interactive to varying degrees. Active input can impact systems design in three ways, namely :

(1) retroactively (or 'after-the-fact' intervention), e.g. human factors audit;
(2) proactively (or 'before-the-fact' intervention or feed-forward prescription), e.g. design handbooks;
(3) enactively (or 'during-the-fact' intervention), e.g. participative design.

Ideally, human factors toolsets should support design actively to ensure the timeliness, relevance and granularity of human factors contributions relative to design support needs at various stages of the design cycle. In other words, one needs to consider :

(1) *what* human factors inputs are necessary to support end-users of the target system;
(2) *how* and *when* human factors inputs should be presented during system design, i.e. software engineers should be supported appropriately as users of human factors design inputs. Better understanding and conceptual cohesion may then accrue. Thus, the uptake of human factors contributions (and their eventual realisation in the design artefact) would improve significantly if human factors inputs are contextualised to existing software engineering conceptions of the system design cycle.

Consequently, the *process* (how and when) and *product* (form and content) of design input are equally important concerns for improving the uptake of human factors contributions during system design. In other words, attention should be directed both at incrementing human factors design knowledge (to widen its scope) and at improving its delivery and presentation.

Bearing the above considerations in mind, an assessment of existing classes of human factors toolsets follows.

The first class of human factors toolsets comprises design principles, guidelines and standards. The design support provided by this class is predominantly declarative (see Williams, 1989; Smith and Mosier, 1984). Although the declarative content varies, design guidelines in general tend to be detailed and lengthy, and their focus can be either vague or specific. Similarly, design principles tend to be simple, compact and general (McKenzie, 1988; Norman, 1988). In addition, design recommendations may vary from mere advice to legislated standards.

Problems reported with this class of human factors toolsets are as follows :

(a) poor scope and presentation format of inputs. Human factors inputs by these toolsets are presented at either too high or too low a level of description. Thus, the recommendations are either too vague to provide effective guidance, or too rigid and specific for application across various design scenarios respectively (Smith, 1986; Chapanis and Budurka, 1990; Hirsch, 1984). In either case, extensive interpretation, adaptation and extrapolation would be required for application during design (Klein and Brezovic, 1986). These problems highlight a serious weakness of the class of toolsets. To aggravate matters, Smith (1986) reported that designers who are insufficiently competent in human factors may not even recognise the need to adapt the declarative content;

(b) conflicting advice may be offered. As the declarative content of these toolsets becomes more comprehensive and detailed, contradictory advice may arise (Maguire, 1982; Alexander, 1987). Generally, no guidance is provided by the toolsets on how such contradictions may be resolved (Marshall, 1984);

(c) poor accessibility of human factors information. As a class, these toolsets are difficult to use (Rogers and Pegden, 1977; Eason and Cullen, 1988), and the reference document quickly becomes voluminous and

daunting (Norman, 1988).[14] Thus, the toolsets tend to become part of the back-drop of design and are therefore overlooked frequently (McClelland, 1990);

(d) questionable validity of human factors recommendations. In particular, the validity of standards is doubtful as current human factors knowledge may not adequately support their imposition (Smith, 1986). Furthermore, existing human factors standards are not testable (Chapanis and Budurka, 1990). Thus, effective enforcement is contentious;

(e) poor mapping of human factors inputs to the design context. Reports have indicated that system developers frequently fail to make relevant human factors considerations at appropriate stages of the design cycle (McKenzie, 1988; Smith, 1986). In response to the problem, a common exhortation is that human factors advice should be sought earlier in the system design cycle. However, in the absence of an explicitly structured framework for human factors design,[15] judgements on what constitutes appropriate and early human factors input remain subjective;

(f) the declarative emphasis of the class of toolsets encourages a narrow view of the recruitment of human factors designers, i.e. as consultants as opposed to active participants in design specification. Comprehensive analysis of human factors concerns may be discouraged as a result;

(g) inadequate project resourcing for a comprehensive account of relevant standards, guidelines and principles. Such difficulties arise because an appropriate representation of human factors design by the system design agenda remains unaddressed.

For these reasons, this class of toolsets has been viewed by designers as restrictive or mere formalities at best, and at worst a hindrance to design (Smith, 1986).

---

[14] For example, Smith and Mosier (1984) identified more than 600 guidelines for user interface design alone.

[15] At the highest level, a structured human factors design framework may comprise a human factors design life-cycle, while at the lowest level it assumes the form of a structured analysis and design method.

In summary, this class of toolsets fail to address adequately the *process* and *product* of human factors input. For instance, when and how inputs should be delivered (process); and what inputs should be recruited in particular system design contexts (product). To resolve these problems, enhancements of this class of toolsets would involve addressing the following :

(1) contextualising the configuration and declarative content of the toolsets appropriately to the design support required at various stages of the system design cycle. To this end, a structured framework for human factors design needs to be defined explicitly. Thus, the scope of human factors guidelines has been extended to include procedural guidance, e.g. BS 6719:1986 (see also Gould, 1988; McClelland, 1990). The format and declarative content of human factors guidelines, principles and standards may then be configured to meet design support requirements as defined by the stage-wise scope and process of the framework. An example of work in this direction may be found in Esgate, Whitefield and Life (1990);
(2) ameliorating problems of accessing a rapidly increasing declarative knowledge-base, i.e. passive delivery through handbooks is an inadequate solution. To address this problem, Smith (1986) suggested the following procedures for identifying relevant design guidelines :

(i) begin with a guidelines review;
(ii) discard guidelines which are irrelevant;
(iii) modify, expand and weigh the importance of the remaining set of guidelines in anticipation of trade-offs and documentation at later stages of system design.

Although Smith's suggestions are helpful, a guidelines review would remain a daunting task in view of (c) and (g) above. Thus, computer-based tools have been developed to support the application of this class of human factors design support. Computer-based tools will be discussed later.

To conclude, principles, guidelines and standards as a class of human factors toolsets, relies heavily on appropriate consultation during system design. In the

absence of an explicitly structured human factors design framework, major pitfalls may include the following :

(a) continued perception of human factors designers as design consultants as opposed to active participants in design specification;
(b) human factors design would remain a craft practice (see Long and Dowell, 1989);
(c) comprehensive human factors analysis may be discouraged due to inadequate allocation of project resources, e.g. the time set aside for human factors design may be unrealistically short.

The second class of human factors toolsets comprises computer-based tools. As with principles, guidelines and standards, computer-based tools should not be developed in a 'vacuum'. In other words, their configuration should support design requirements at each stage of the system design cycle. Thus, design support requirements must be identified sufficiently if appropriate functional coverage is to be realised in such tools.

Generally, computer-based tools provide both procedural and declarative design support. Procedural support provided by computer-based tools may include the following (after existing Software Engineering tools) :

(a) project planning, e.g. for scheduling design deliverables, tracking project progress;
(b) design specification and documentation, e.g. text and graphics editors, consistency checkers (for notational rules);
(c) design evaluation, e.g. simulators, prototypers, animators;
(d) design implementation, e.g. compilers, linkers.

To provide *'total systems solution'* or *'design cycle support'* the above categories of design support should be covered comprehensively (Hewett and Durham, 1987).

Computer-based tools that provide declarative design support are a recent development. For instance, tools that monitor the appropriate application of human

factors design guidelines are emerging (Perlman, 1987).

In general, computer-based tools suffer from similar problems as principles, guidelines and standards, e.g. Hartson and Hix (1989) reported that the functional coverage of computer-based tools is frequently too narrow. Aside from these problems, it is frequently the case that a long time lag separates the recognition of design support needs and the development of a computer-based tool. For instance, Software Engineering CASE and IPSE tools were not developed until structured analysis and design methods became well established (Hewett and Durham, 1987). In view of these observations, it may be expected that computer-based tools for human factors would not be realised unless its design cycle is defined explicitly. Specifically, appropriate requirements for computer-based design support can not be identified in the absence of an explicit conception of human factors design. Consequently, for human factors the development of computer-based tools may be considered a longer term objective.[16]

The final class of human factors toolsets comprises design techniques and methods (e.g. task analysis). Examples of such toolsets are described by Meister (1984) and Kloster and Tischer (1984). Although there is a wealth of human factors techniques and methods, the procedural support provided by this class of toolsets generally suffer from the following problems :

(a) too narrow a coverage of the system design scope. Existing toolsets tend to focus on later stages of the design cycle;
(b) the format of toolset outputs is contextualised poorly to design support needs at each stage of the system design cycle;
(c) the toolsets are difficult to use (Wilson et al, 1986), and are expensive and time consuming to apply, e.g. rigorous experiments. In addition, the validity of the derived results may be doubtful, e.g. experimental results may not be applicable to real world tasks since they are derived under

[16] Current computer-based support for human factors design is in the early stages of development. The support presently provided is limited, e.g. HUFIT tools (ESPRIT 385, 1989 and 1990) comprise primarily checklists and form-fill schemes.

controlled conditions.

In conclusion, to achieve maximum effectiveness all classes of toolsets should be configured with respect to a better defined human factors design cycle. For instance, the scope of current human factors methods and techniques could be extended to cover the design cycle more completely. Since adequate human factors computer-based tools are generally unavailable at present, the extended methods and techniques would facilitate their development, e.g. their explicit design scope, process and representation constitute design support requirements to be met by the tool (see also CCTA (Draft) Report, 1988, pg. 20; Alvey Human Interface Committee Report, 1987, pg. 18 and 32). In other words, a pre-requisite for enhancing existing toolsets is the definition of a sufficiently complete and structured human factors design framework. To this end, well developed Software Engineering conceptions of the system design cycle should be examined. In this respect, two conceptions have gained prominence, namely those entailed by rapid prototyping and the traditional system development life-cycle (as instantiated by SADMs). A brief review and comparison of the conceptions follows. The focus of the comparison is on the strengths and weaknesses of the conceptions :

(a) for facilitating human factors involvement throughout system design;
(b) in providing a model for configuring a human factors design cycle.

2.2. Conceptions of the System Design Cycle : Rapid Prototyping versus the Traditional System Development Life-Cycle

It was suggested that a pre-requisite for enhancing existing human factors toolsets is a *structured* human factors design framework. By definition, a structured framework is one whose *stage-wise scope, process* and *notation* are sufficiently *explicit* and *complete* with respect to the *system design cycle.* To specify such a framework, the following questions need to be answered :

(a) what conceptions of the system design cycle presently exists;
(b) which conception would best support the derivation of a structured

human factors design framework;

(c) what is entailed by the derivation of such a framework ?

Question (c) will be addressed in Chapters Three and Five.

Questions (a) and (b) constitute the subject of this sub-section. Specifically, two contrasting conceptions are assessed for their potential as a reference base for constructing a human factors design framework. The conceptions assessed comprise rapid prototyping and the traditional system development life-cycle (exemplified by SADMs).

Rapid prototyping involves 'fast-building' a preliminary design followed by various forms of iterative testing and prototyping cycles, namely step-wise or incremental prototyping, evolutionary prototyping and throw-away prototyping (Hekmatpour and Ince, 1987). The essence of rapid prototyping is either a brief or an altogether absent design specification stage. Design documentation at this stage is usually poor.

Rapid prototyping is frequently equated with prototyping (e.g. Wilson and Rosenberg, 1988). For instance, benefits of *prototyping* in general have frequently been cited as arguments in support of *rapid prototyping*. Such assumptions are misleading as these benefits may be reaped without incurring costs which are unique to rapid prototyping. In particular, to speed up the construction of a prototype, rapid prototyping typically minimises the time spent on design analysis and specification. Thus, the assumption of rapid prototyping (implicitly or otherwise) is that iterative prototype construction-and-test cycles constitute an adequate substitute for such a design phase. It can not be over-emphasised that the assumption is fallacious as prototyping a design is not the same as designing a prototype (the former may involve minimal analysis and design). In this respect, the time saved by skimping on the design phase may subsequently incur heavy costs in terms of extensive maintenance, updates and bug fixing (Shuttleworth, 1987). Alternatively, an uneconomic number of prototypes may have to be tested before a satisfactory solution is eventually derived (Long and Neale, 1989; Keller, 1987). In contrast with the assumption of rapid prototyping, prototyping is usually

undertaken *following* comprehensive design analysis.[17] Consequently, prototyping and rapid prototyping should be differentiated explicitly. Further differences will be highlighted later when rapid prototyping is compared with SADMs. To support these considerations, the merits and demerits of rapid prototyping are listed in Figures 2-2 and 2-3 respectively.

In the context of a traditional system development life-cycle, design is described in terms of sequential phases, each of which implicates a number of stages. An early example of the life-cycle conception is the waterfall model (see Bóhm, 1984; Jensen and Tonies, 1979; Fox, 1982). Generally, existing conceptions of the traditional system development life-cycle differ essentially in the scope of their design coverage and the number of phases into which the scope is decomposed, e.g. the design scope may range from business analysis (Wigander et al, 1979) to post-mortem examination of retired systems (Olle, 1988); and the number of phases may vary from three to fifteen phases. Although slight differences in the sequencing of design phases exist, there is general agreement on the conceptual aspects of system design, e.g. the emphasis on : requirements before specification,

Figure 2-2 : Merits of Rapid Prototyping

Arguments for rapid prototyping include the following :

(a) it enables earlier conceptualisation of the design problem and facilitates the elicitation of user feedback;
(b) by demonstrating the prototype, it helps users visualise their system requirements. A more accurate identification of user requirements may thus be derived. Consequently, the quality and completeness of the functional specifications is improved, and the probability of attaining the expected performance is increased;
(c) it reduces development time and encourages the investigation and testing of various design solutions;
(d) it provides a tangible design artefact for problem analysis and discussion among design team members;
(e) it provides a means of testing design concerns specific to the artefact.

[17] Prototyping unlike, rapid prototyping, is thus compatible with structured analysis and design methods, e.g. see Multi-User Computing, (1989), pg. 26.

Figure 2-3 : Demerits of Rapid Prototyping

Arguments against rapid prototyping include the following :

(a) two unsatisfactory outcomes may arise if inadequate time has been spent on analysis and problem formulation, namely : an inappropriate prototype may be constructed; and incorrect test criteria may be applied to the prototype leading to inappropriate interpretations of test results. Thus, the design rationale used in the construction of subsequent prototypes may be flawed. Consequently, successive iterations may not facilitate efficient convergence onto a design solution;
(b) inappropriate prototypes may cause end-users to be committed prematurely to a specific design solution. As a result, inadequate problem analysis and poor design solution may be exposed at too late a design stage (Long and Neale, 1989). Thus, Thimbleby (1987) highlighted that rapid prototyping potentially violates the principle of delaying design commitment;
(c) there may be resistance towards discarding prototypes especially if the time and expense incurred in their creation is not insignificant. Thus, non-ideal design characteristics may be carried forward. In the worst scenario, prototypes may be passed off as the final system (Boar, 1984; Fox, 1982);
(d) design audits can not be conducted properly as the design documentation is frequently inadequate. Thus, rapid prototyping may propagate design communication problems similar to program 'hacking'. Inadequate documentation would also imply poor support for later design modification and maintenance;
(e) a heavy reliance on computer-based tools can lead to over-design (Mantei, 1986). In addition, the paradigm implied by rapid prototyping of entering knowledge directly into the computer-based prototype, engenders an ad hoc or 'magic box' strategy. Such a strategy would not adequately support projects with ill defined domains, and co-operative work in design teams that characterises large system development (Long and Neale, 1989);
(f) rapid prototyping encourages inappropriate deferment of design decisions since it relies heavily on their resolution via prototype construction-and-test cycles (Grudin et al, 1987);
(g) limitations and constraints that apply to the target design artefact can be ignored during prototyping;
(h) a prototype can be oversold, creating unrealistic expectations in the target design artefact;
(i) the prototyping process can be difficult to manage and control.

specification before implementation; design iterations; etc.

Despite repeated criticisms and calls for their rejection, life-cycle conceptions have persisted to the present. The conceptions survived because the criticisms are either not serious, or may be attributed to expectations that are incongruent with its intended purpose (see later). Some of the criticisms are reviewed below.

The most common criticism is that life-cycle conceptions describe system design erroneously as proceeding in *discrete* sequential phases. Overlaps between adjacent phases have been cited as evidence to the contrary (Grudin et al, 1987). Although this criticism may be correct, it should be noted that life-cycle conceptions are not

intended as accurate models of system design activities. Instead, they are simplified design frameworks that support and so facilitate the systematic management of design tasks.

Another criticism of life-cycle conceptions is that system development is viewed as *a discrete event* with defined start- and end-points (Grudin et al, 1987). Although this is a fair criticism, one could view such conceptions as describing a steady-state, finite element with implicit inputs from preceding projects and influences on future projects.

A further criticism of the validity of life-cycle conceptions argues that system design is not sequential but parallel. Multiple asynchronous threads of modular design development are cited as evidence. Nevertheless, it may be argued that such observations do not necessarily imply the breakdown of the life-cycle model since design in each of the threads may still conform to the latter model. In other words, modular design development is not excluded by life-cycle conceptions.

Consequently, such arguments do not affect significantly the utility of life-cycle conceptions for supporting system design. Indeed life-cycle conceptions have persisted through the years, and design methods have been established on their basis. The most notable of these methods is a class commonly referred to as structured analysis and design methods (SADMs). A brief account of these methods follows.[18]

SADMs, as a class of Software Engineering methods, are defined by the following characteristics :

(a) the procedural support they provide is reasonably complete with respect to the system design cycle;
(b) system design is advanced in stages, each of which comprise explicitly

[18] Formal methods will not be considered here since they do not generally cover the system design cycle sufficiently, e.g. requirements capture is unlikely to be a target of a formal method (Norris, 1985).

defined scope, process and notation, i.e. design is advanced via a series of intermediate design products;
(c) independent design concerns are addressed separately, e.g. analysis before specification, specification before implementation.

Having defined what constitutes a SADM, existing misconceptions about these methods should be dispelled. Major misconceptions comprise the following :

(a) the design approach of SADMs, as a class, involves 'getting-it-right-the-first-time' (Gould and Lewis, 1983). Such a view has been interpreted to imply that SADMs do not involve iterative design. Thus, a more appropriate characterisation of the design approach of SADMs is 'getting-it-right-to-begin-with' (Grudin et al, 1987) or 'getting-the-first-best-guess-solution';
(b) all SADMs involve top-down design exclusively. The generalisation can not be true for two reasons, namely :

(i) design is seldom confined to top-down processes only;
(ii) some SADMs have explicitly disowned top-down design and emphasised on a 'middle-out' approach instead, e.g. JSD.

Thus, criticisms directed at top-down design should not be applied unquestioningly to SADMs. Unfortunately, sweeping criticisms have been proposed on this basis, e.g. the design support capability of SADMs is confined only to well defined domains;
(c) prototyping does not have a role in SADMs. This fallacy is addressed later when SADMs are compared with rapid prototyping. To this end, the merits and demerits of SADMs are listed in Figures 2-4 and 2-5 respectively.

A comparative assessment between SADMs and rapid prototyping follows. It is argued presently that SADMs constitute better reference frameworks for the specification of an explicit human factors design cycle. A structured method may then be developed for human factors to support a more effective incorporation of its

inputs throughout system design. The arguments for SADMs are three-fold.

Firstly, the benefits of rapid prototyping are shown to be matched potentially by

Figure 2-4 : Merits of Structured Analysis and Design Methods

Existing Structured Analysis and Design Methods (SADMs) provide the following design support :

(a) quality assurance : the well defined stage-wise scope and process of SADMs facilitate design reasoning and decisions. In addition, the methods encourage complete logical analysis before physical design (Hares, 1987). Also see other points below;
(b) management of design complexities : the nature and sequence of intermediate design decisions are defined explicitly by SADMs. In addition, independent concerns are separated, while related concerns are grouped appropriately;
(c) design communication : well developed stage-wise notations of SADMs support a range of intermediate design descriptions which facilitate communication between system developers, between developers and managers, and between developers and users (Hares, 1987);
(d) review of design decisions and rationale : design audits are supported by the explicit and comprehensive design documentation advocated by SADMs;
(e) continuous verification and validation of design specifications : the emphasis of SADMs on the production of stage-wise design products encourages testing, prototyping, and design iteration at various stages of the design cycle;
(f) detailed project planning : intermediate design products of SADMs are defined explicitly. Thus, design resource estimations and the setting of project milestones are both supported;
(g) recruitment of inter-discipinary knowledge and methods : the well developed methodological structure of SADMs constitutes a framework for recruiting such knowledge and methods. Thus, missing or inadequate design knowledge may be identified for further attention.

Figure 2-5 : Demerits of Structured Analysis and Design Methods

General criticisms of existing Structured Analysis and Design Methods (SADMs) include the following :

(a) they are deficient in the identification of user requirements;
(b) they do not address user interface design;
(c) their notation may not adequately convey the actual workings of the system to the user (Mantei and Teorey, 1988). However, it should be noted that positive reports on these notations have also been published, e.g. Hares (1987);
(d) they require comprehensive intermediate design products and documentation. These requirements exact heavy demands on resources that are not available to smaller system development projects;
(e) their application may be cumbersome unless supported by computer-based tools.

SADMs. In particular :

(1) advantages (a) and (b) of rapid prototyping (i.e. facilitating design discussions and user feedback elicitation, Figure 2-2) are also supported by the explicitly defined and documented stage-wise design products entailed by SADMs. Prototyping is similarly encouraged by SADMs (Long and Neale, 1989; Essink, 1988; Keller, 1987). Thus, remaining doubts on the efficacy of SADM notations for facilitating user feedback may be removed by the emphasis on prototyping and the use of graphical notations for design description (see Fitter and Green, 1979). In addition to these arguments, it should be noted that rapid prototyping may not be viable for novel design (Thimbleby, 1987). Specifically, an adequate start-point is required before a design can be prototyped, i.e. an extended period of analysis and design is necessary. Such design scenarios are supported by SADMs;
(2) advantage (c) of rapid prototyping (i.e. significant reduction in system development time, Figure 2-2) is questionable for the following reasons :

(i) demerits (a) and (d) in Figure 2-3;
(ii) design management problems inherent in rapid prototyping (Crinnion, 1989);
(iii) difficulties in integrating final design specifications (Bóhm, 1984; Morrison, 1988).

Thus, failures of rapid prototyping in realising faster system development have been reported, e.g. Grønbák (1989) that in nine projects undertaken using rapid prototyping, deadlines for design completion were all exceeded considerably. As a result, overall development times were found to be the same as other system design approaches.

Secondly, critical human factors reservations in respect of rapid prototyping should be considered. Specifically, rapid prototyping may not be compatible with human

factors objectives because :

(a) rapid prototyping may be seen as *replacing* rather than *supplementing* early human factors involvement in system design (Grudin et al, 1987; Clark and Howard, 1988). For instance, it may restrict its involvement to prototype evaluation as opposed to active design specification. This problem is serious since prototyping would not substitute adequately for design (see Figure 2-3 : (a), (b) and (f));

(b) rapid prototyping engenders a false impression that human factors is contributing effectively to system design. In the best case, the recruitment of human factors only at the prototype evaluation stage, may propagate the 'too-little-too-late' problem of human factors input to system design. In the worst case, resistance against discarding an expensive prototype (see (c) in Figure 2-3) may result in non-implementation of human factors recommendations;

(c) rapid prototyping engenders a false impression that end-users are invariably involved in the test and modification of successive prototypes. Reports indicate that prototype evaluation and modification may be undertaken exclusively by expert reviewers (Long and Neale, 1989).

Thus, it is concluded that a number of critical human factors problems remain unaddressed by rapid prototyping. Two particularly serious concerns comprise the absence of a direct human factors contribution to design analysis and specification, and the frequently inadequate documentation of design decisions and rationale. The importance of adequate design documentation is highlighted by the meteoric rise of reverse engineering, design recovery and re-documentation methods in Software Engineering.[19] These design recovery methods exact high resource costs on the project in both human and financial terms. Thus, SADMs are increasingly applied in system development. Consequently, SADMs constitute better reference

[19] These methods had to be invoked frequently to rectify design inadequacies (Chikofsky and Cross II, 1990).

frameworks for structuring human factors input to system development.[20]

To complete the argument for SADMs, a case for adopting SADMs over a simple system development life-cycle should be made. To this end, the characteristics of SADMs should be assessed with respect to the provision of more specific contexts for structuring human factors inputs throughout the system design cycle. These concerns are addressed in the next sub-section.

2.3. A Case for Integrating Human Factors with Structured Analysis and Design Methods

Having argued against rapid prototyping as the reference framework for structuring human factors inputs, a further question needs to be answered, namely : do SADMs provide better reference frameworks than the traditional system development life-cycle ? In other words, what additional benefits motivate the adoption of SADMs instead of a simpler life-cycle conception ? Thus, the contrast is between the following schemes for structuring human factors inputs throughout the system design cycle :

> I) taking the reference framework defined by the stages of the traditional system development life-cycle, intersecting Software Engineering and Human Factors design concerns may be identified. Relevant human factors contributions may then be located as appropriate. Thus, existing human factors toolsets may be 'clustered' as 'toolkits' around specific stages of the system design cycle. This assignment of existing human factors toolsets with respect to the traditional system development life-cycle was previously

[20] A choice must be made between the two conceptions of the system design cycle because rapid prototyping and SADMs are incompatible design approaches (note that SADMs are compatible with prototyping in general -- see Footnote 17). In particular, SADMs emphasise a stage-wise design analysis, specification and documentation as opposed to the rapid generation of a prototype. Thus, the time and effort spent on design specification by the two approaches are very different.

proposed by Grandjean (1984); Berns (1984); Rubinstein and Hersh (1984); Gould (1988); McClelland (1990); Meister (1984); Eason (1987);

II) taking the reference framework defined by explicit characteristics of a particular structured analysis and design method (comprising stage-wise scope, process and notation), intersecting Software Engineering and Human Factors design concerns may be identified. With respect to the characteristics and support requirements of the SADM, existing human factors methods are recruited and then organised into a corresponding structured human factors method.[21] Since both Software Engineering and Human Factors methods are structured explicitly, their integration is facilitated. Methodological integration entails inter-weaving design stages, products and notation (if possible) entailed by the methods of the individual disciplines. Thus, the scope, timing and communication of human factors inputs are contextualised to specific design support requirements of the chosen SADM.

A brief assessment of the above schemes follows.

Although scheme I contributes to the solution of some human factors problems, others remain unaddressed. For instance :

(1) inadequate allocation of project resources for human factors design. The allocation deficit may be attributed to the absence of a sufficiently structured conception of human factors design. Such problems have been reported widely, e.g. Meister (1984); Chapanis and Budurka (1990); Pikaar et al (1990). Although the inclusion of a human factors engineer at the project planning stage may alleviate the problem, accurate prediction of resource requirements (e.g. development time and effort) would still be difficult in

[21] The structured human factors method is expected to provide a reasonably complete coverage of the system design cycle. Human factors design is expressed by the method in terms of explicitly defined stage-wise design scope, process and notation. On the basis of such a method, other human factors toolsets, e.g. declarative and computer-based toolsets, may then be recruited as appropriate.

the absence of comprehensive records of previous projects. To obviate the problem, O'Niel (1980) reported that records gathered with respect to a structured conception of system design activities are instrumental for good project planning. In other words, the collation of such records is supported by structured methods. Thus, scheme II would facilitate an appropriate accommodation of human factors design needs by the system design agenda;

(2) problems in integrating design descriptions since the format and notation of outputs from individual human factors techniques and methods may be very different;

(3) problems in communicating human factors design outputs to software engineers since human factors notations are frequently insufficiently specific;

(4) sub-optimal uptake of human factors contributions. In the absence of an explicit conception of how human factors design is structured within the system design cycle, the scope, timing and granularity of human factors contributions may not be specific to the system design context. The uptake of human factors outputs is thus affected adversely;

(5) inefficient utilisation of project resources. Following the identification of suitable human factors techniques and methods, care must be taken to avoid unnecessary repetitions during their individual application. In other words, design activities of disparate methods which overlap have to be concatenated. Information requirements should also be reconciled across the methods. These pre-requisites are common. For instance, human factors methods have been reported to indicate only what should be designed but not how, e.g. prescriptive advice on how design decisions may be formulated following initial analysis is frequently omitted (Sutcliffe, 1989; Chapanis and Budurka, 1990). Thus, in the absence of a structured human factors method, valuable project time would have to be spent on :

(a) assessing the suitability of individual human factors methods;

(b) tailoring relevant methods into a coherent set as described above;

(c) extending the methods to ensure sufficiently structured and

complete coverage of the system design cycle, e.g. human factors methods may be inadequate in stage-wise scope, process and notation.

To perform the above tasks, the human factors designer would have to be very well informed.[22]

In summary, four requirements for improving human factors input to system design are highlighted by the above account, namely :

(a) the coverage of human factors toolsets should be sufficiently complete with respect to the system design cycle;
(b) human factors toolsets should be coherently integrated;
(c) resource requirements for human factors design should be accommodated explicitly by the overall design agenda;
(d) Human Factors and Software Engineering design inter-dependencies should be defined sufficiently throughout the system design cycle.

Presently, scheme II (i.e. methodological integration via SADMs) is reviewed to highlight its potential as a solution that satisfies the above requirements.

Since their introduction in the seventies and eighties, SADMs have made rapid inroads into system development. For instance, SSADM has been recommended as a standard by the CCTA (Hewett and Durham, 1987). Notwithstanding the above account, such events may necessitate the following appraisal of human factors involvement in system development :

(1) the impact of SADMs on the effectiveness of current means of human factors input. For instance, are existing human factors methods compatible with SADMs ? Since these concerns are subsumed in (2) below, they will be discussed when the case for the integration of Human Factors and

[22] This task may not be trivial. For instance, a recent report counted 96 methods for task analysis alone. Thus, the need to perform such tasks while under project pressures may be unacceptably inefficient (as well as stressful).

Software Engineering methods is presented;
(2) the possibility of exploiting SADMs for improving the effectiveness of current human factors input. For instance :

(a) the explicit stage-wise design scope, process and notation of SADMs may comprise methodological characteristics to be emulated by human factors methods;
(b) the well defined design framework of SADMs constitutes a reference context for structuring human factors input to system design. The timeliness, granularity, and format of human factors inputs vis-a-vis design support needs at various stages of the system design cycle is thus ensured.

Exploiting SADMs in this way, constitutes a structured integration of Human Factors and Software Engineering methods, i.e. scheme II above.

A more detailed account of the arguments in favour of scheme II follows :

(1) well developed notations of existing SADMs may be used to describe human factors design outputs. The recruitment may help to improve the specificity of human factors descriptions. In addition, the use of a common notation may also facilitate the communication of human factors design products to software engineers;
(2) the emphasis of SADMs on comprehensive design documentation facilitates the capture of design rationale, i.e. antecedents and consequents of design decisions are documented explicitly. Comprehensive documentation is essential to support quality assurance since it is an effective means for detecting and correcting design errors, e.g. design audits (and hence input) by human factors designers and end-users are supported appropriately (see Long and Neale, 1989; Butler et al, 1989; Akscyn and McCracken, 1984; Alvey Human Interface Committee Report, 1987, pg. 34). The recruitment of documentation schemes of existing SADMs is also important because adequate design records are frequently

written into contractual agreements;
(3) the well defined characteristics of SADMs support a more specific intersection with human factors design concerns, i.e. SADMs provide a more specific framework for locating human factors contributions. For instance, existing human factors methods may be recruited to meet design support requirements specific to the chosen SADM. The recruited methods may then be developed into a structured human factors method. Such a method benefits human factors in three ways.

Firstly, it facilitates the identification of suitable human factors toolsets for further development. For instance :

(a) deficient areas of declarative human factors knowledge may be highlighted for further research. In the mean time, user testing on prototypes corresponding to design products of the method may compensate for such knowledge deficiencies;
(b) computer-based tools comparable to Computer-Aided Software Engineering (CASE) tools and Integrated Project Support Environments (IPSEs) may be developed based on the structured human factors method (see Hewett and Durham, 1987; Bott, 1988 for an account on the contributions of SADMs to the development of such computer-based tools).

Secondly, it facilitates the recruitment of relevant declarative human factors knowledge.

Thirdly, Human Factors and Software Engineering design roles and inter-dependencies may be specified explicitly by integrating a structured human factors method with the chosen SADM. In other words, human factors design stages are located explicitly against corresponding SADM design stages. Such an integration of the methods would promote greater awareness and understanding of inter-disciplinary design needs. For instance, integration would provide familiar reference points that support the assimilation of human factors inputs by software engineers (and vice versa).

Thus, more effective human factors input and uptake may be expected;

(4) the structured integration of human factors methods and SADMs constitutes the specification of a 'unified' system design cycle. Such a design cycle would ensure an explicit accommodation of human factors design needs. Specifically, adequate design resource allocation may be ensured in two ways :

(a) better project planning and estimates of resource requirements. In particular, more accurate projections of system development timescales would accrue from an explicit definition of inter-disciplinary design deliverables. Projections of design resource requirements are also supported by more specific records of previous projects.

(b) explicit representation of human factors on the design agenda. In other words, resource are allocated explicitly for human factors design. Thus, encroachment on its design resources may be obviated. Explicit resource allocation is important because encroachments have been reported frequently. For instance, unrealistic time schedules have been imposed on human factors design (often considered a lower priority) as a result of technical difficulties and delays in system launch (Eason and Cullen, 1988; Meister, 1984).

For these reasons, the focus of the present research is on the integration of structured human factors methods with SADMs. Presently, the preliminary research scope and objectives described in Chapter One may be detailed further.

2.4. Detailing the Scope and Objectives of the Research

The research scope comprises the derivation of a general conception of structured human factors design, followed by its instantiation as a structured method that is appropriate for integration with the chosen SADM. To this end, the objectives of

the research comprise the following :

(a) to specify general requirements and a research scheme for integrating structured human factors methods with SADMs (see Chapter Three);
(b) to derive a general conception of structured human factors design by collating and extending existing conceptions (see Chapter Four);
(c) to review previous research on the integration of human factors methods with SADMs (Chapter Five);
(d) to describe research concerns involved with the specification and subsequent integration of a structured human factors method with a chosen SADM (namely the Jackson System Development (JSD) method). The specification involves instantiating the general requirements and research scheme in (a) above with respect to both the JSD method and specific constraints of the research (e.g. available resources may limit the scope of the research to a subset of (b) above). These concerns are described in Chapter Six;
(e) to construct an appropriate structured human factors method for integration with the JSD method. Method construction and integration should satisfy the requirements stipulated in (a) and (d) above. In addition, lessons learnt in (c) above should be incorporated (see Chapters Seven to Eleven);
(f) to assess the research in general and the integrated method in particular; and to propose directions for furthering the present research (see Chapter Twelve).

A detailed account of the work entailed by these objectives follows.

PART II :

On Human Factors Integration with Structured Analysis and Design Methods (SADMs)

CONTENTS

Chapter Three : General Research Requirements, Activities and Plan for Integrating Human Factors with SADMs.....61

3.1. General Requirements of Methodological Integration.....61
3.2. General Research Concerns for Methodologically Integrating Human Factors with SADMs.....65
3.3. A General Research Plan for Methodologically Integrating Human Factors with SADMs.....71

Chapter Four : Towards a Conception of Structured Human Factors Design.....78

4.1. A Survey of Existing Conceptions of Human Factors Design.....79
4.2. Analytic Derivation of a Simplified Framework for Structured Human Factors Design.....89
4.3. Towards an Enhanced Conception of Structured Human Factors Design.....99

Chapter Five : A Review of Previous Research into the Integration of Human Factors with SADMs.....105

5.1. Human Factors Integration with the Structured Analysis and Structured Design (SASD) Method.....105
5.2. Human Factors Integration with the Structured Systems Analysis and Design Method (SSADM).....114
5.3. Human Factors Integration with the Jackson System Development (JSD) Method.....120
5.3.1. The Work of Sutcliffe (1988a and b).....120
5.3.2. The Work of Carver, Clenshaw, Myles and Barber (1987).....124

Chapter Three : General Research Requirements, Activities and Plan for Integrating Human Factors with SADMs

"The last thing one knows in constructing a work is what to put first."

*Blaise Pascal, 1909, **Pensées**.*

Observations made in the preceding chapters are drawn together in this chapter to characterise general research requirements for structured integration of human factors methods with SADMs. Thus, the concerns of the present chapter are as follows :

(a) defining what constitutes structured integration of human factors methods with SADMs. The definition serves two purposes, namely it identifies high level requirements for integration, and sets the criteria for assessing previous research in the area (see Chapter Five);
(b) specifying a general research plan for achieving structured integration. This specification serves two purposes. Firstly, it provides general guidance to other researchers who may wish to undertake similar work. Secondly, its framework supports the formulation of a specific research plan for instantiating structured integration of human factors methods with a chosen SADM (see Chapter Six).

These concerns are expanded in the following sub-sections.

3.1. General Requirements of Methodological Integration

The most general requirement for structured integration relates to the functional definition of a method. Most definitions are expressed in terms of the design scope, process and notation of a method (e.g. Carver, 1988; Maddison, 1983; etc.). For

instance, the following definition was suggested by Hartson and Hix (1988) :

> *"A methodology for system development consists of a set of procedures that indicate a step-by-step development process over a life-cycle, and a notational scheme that is the means for representing designs that evolve during that life-cycle."*

SADMs, as a class of methods, requires additionally that the stage-wise :

> (a) design scope should be well defined;
> (b) design process should be sufficiently explicit, grouped appropriately and sequenced logically to support system development;
> (c) design notations should be appropriate for documenting design products which are generated at various stages of the system design cycle.

Thus, it follows that a structured integration of human factors methods with SADMs should uphold the above methodological requirements. Specifically, general requirements for methodological integration comprise the following :

> (1) design stages of the structured human factors method should be intersected appropriately with those of the SADM, i.e. the design scope and process of the two methods are inter-woven to define a coherent overall design schedule;
> (2) existing notations of the SADM should be recruited (with extensions as necessary) for describing human factors design products. The maximum use of a common notation supports inter-disciplinary design communication. In particular, it improves the communicability of human factors design products;
> (3) tighter design relationships should be defined between the structured human factors method and the SADM to facilitate overall design management, e.g. to ensure efficient co-ordination of design team

activities.[1] Thus, the status of serial and parallel design stages between the two component methods should be made more explicit. Since serial design stages imply information inter-dependencies, obligatory contact points (termed design inter-dependencies) should be imposed. In the case of parallel design stages, check points should be specified to ensure efficient design.[2] Specifically, check points (or design conjunctions) are necessary to ensure that design specifications generated by the two streams of the integrated method are convergent, i.e. to check 'design drift'. Thus, check points should be specified for stages where much design extrapolation is involved. Too few check points or the failure to adhere to them, may result in potential mismatches between Software Engineering and Human Factors designs. Additional design iterations are necessary and the design thus becomes inefficient.

The above requirements for methodological integration may be met in three ways, namely by :

(i) directly integrating structured human factors methods and SADMs;
(ii) extending a structured human factors method to include a SADM that has been configured to suit the characteristics of the former;
(iii) extending a SADM to include a structured human factors method that has been configured to suit the characteristics of the former.

Although there is a wealth of human factors methods (Gould, 1988), none of the methods could be considered to provide a sufficiently structured coverage of the system design cycle (see Chapters One and Two). Consequently, (i) and (ii) above

---

[1] Large system development projects are generally undertaken by multi-disciplinary design teams. Thus, it is reasonable to assume that human factors designers and software engineers would undertake design with respect to the structured human factors method and SADM respectively. An integrated method should therefore specify how the disparate design streams should be co-ordinated.

[2] Situation-specific check points (e.g. those peculiar to the organisation or design team) should be excluded to maintain the general applicability and flexibility of the method.

are not directly amenable routes for a structured integration of human factors methods with SADMs. Conversely, since the stage-wise design scope, process and notation of any SADM is better formed than any comparable human factors method, (iii) represents a logical route for structured integration. Other arguments for selecting (iii) are as follows :

(1) the current status of SADMs -- SADMs are already well established. As such, the integration should focus on how human factors methods may be configured to support existing Software Engineering practice;
(2) existing tool support for SADMs -- computer-based support tools are available. Such tools (e.g. graphics editors) may benefit human factors design if SADM notations were adopted for describing its design products;
(3) the current role of human factors in system design -- since human factors plays a supporting role, the onus is on human factors to ensure that its methods are contextually appropriate to support early and continued system design involvement (Carver, 1988). In addition, appropriate support should be provided to software engineers to ensure an effective assimilation of human factors into the existing system design practice. Thus, human factors methods should be located against reference points with which software engineers are already familiar, e.g. the design stages of SADMs.

However, a price is paid in structuring human factors methods around SADMs. Specifically, a constraint is imposed which requires existing SADMs to be left largely unchanged by methodological integration. Since SADMs are generally well developed, the consequences of the constraint would not be severe if human factors design is represented appropriately in the integrated method (see later).

Presently, the research following (iii) is described. Specifically, (iii) implies the construction of a structured human factors method that satisfies the design support required by the chosen SADM. To this end, general research requirements would comprise the following :

(a) extension of the design scope of the SADM to include human factors.

The extension is not necessarily a simple addition since the scope covered by a particular SADM may overlap with human factors, e.g. requirements analysis;

(b) integration of human factors design stages with corresponding SADM stages. In other words, the location and timing of human factors design processes are specified with respect to those of the SADM. Design inter-dependencies are thus identified;

(c) extension of the notational capability of the SADM to include the description of human factors products. As in (a) above, the extension is not necessarily a simple addition since SADM notations may be sufficiently powerful for describing human factors products.

These research requirements may be matched against the pre-requisites for effective human factors input (see Chapter Two). In particular, human factors contributions should be matched against design support needs at various stages of the system design cycle. Thus, the scope, timing, granularity and format of human factors products are appropriate for the system design context. Improved uptake of human factors is thus ensured by the greater applicability, relevance and communicability of its stage-wise design inputs.

An account of research concerns that correspond to the above requirements follows.

3.2. General Research Concerns for Methodologically Integrating Human Factors with SADMs

It was suggested earlier that methodological integration implicates the specification of a structured human factors method that appropriately supports the design context of the chosen SADM. General research requirements for integration were then identified. The requirements may be detailed into the following research concerns :

(a) derivation of a conception of structured human factors design. The conception should be constructed by extending existing conceptions;

(b) identification of human factors design support required by the SADM.

The required support may be identified by intersecting the design scope of the SADM with the conception derived in (a) above. Design support requirements should be identified for individual design stages of the chosen SADM. Thus, a pre-requisite is an adequate understanding of the SADM;
(c) specification of a structured human factors method to support the chosen SADM. Specifically, a review of existing human factors methods may be conducted with respect to the design support requirements identified in (b) above. For instance, existing requirements analysis methods would be reviewed if the scope of the SADM does not explicitly include such design considerations. Relevant methods are then modified and recruited to the structured human factors method that is being constructed. The recruitment of existing methods is motivated by two reasons, namely :

(i) faster convergence on an acceptable structured human factors method may result since it builds upon existing human factors knowledge;
(ii) positive transfer of training with respect to the subsequent application of the structured method may be maximised for the same reason.

Prior to their recruitment, existing methods may be developed further to meet the requirements for methodological integration. For instance, the design scope of the methods may be extended and their stage-wise design products, process and procedures may be defined more explicitly;
(d) integration of the structured human factors method with the chosen SADM. Following the derivation of a satisfactory[3] structured human factors method, its integration with the SADM may be specified explicitly. At this juncture, three requirements of methodological integration should be addressed (see previous sub-section). Specifically :

(i) the design stages of the structured human factors method should be

[3] The reader is referred to Chapter Six for an account of what constitutes a *satisfactory* structured human factors method.

inter-woven appropriately with those of the SADM;
(ii) existing notations of the SADM should be recruited (as appropriate) for describing human factors design products. Note that this requirement may be addressed initially in (c) above;
(iii) contact points (termed design inter-dependencies) and check points (or design conjunctions) should be specified to ensure efficient design.

(e) evaluation and iterative development of the structured human factors method and the integrated method.[4] Iterative development may be applied following each method specification cycle.

More needs to be said about method tests alluded in (c) to (e) above. In configuring method evaluation tests, two considerations should be addressed, namely :

(I) the tests should take account of the current status of the methods. In the present context, both the structured human factors method and integrated method are under development. Thus, the tests should be configured to provide information that would support iterative method development.
(II) the tests should consider resource constraints of the research. In particular, its configuration should allow a sufficient number of test cycles without the need to encroach on resources allocated to address other research concerns.

To satisfy consideration (I), the tests should address the following issues :

(a) *demonstration* of the design support provided by the methods at each stage of system development;
(b) *validation* of the methods in the field to establish the pertinence of the design support provided with respect to a real design context;

[4] The focus of integrated method development is confined to the specification of appropriate design contact- and check-points, since its component SADM remains essentially unchanged.

(c) *realisation* of superior design artefacts. Specifically, a link should be established between the derivation of superior design artefacts and improvements in human factors design support attributable to methodological integration.

Although the issues are interacting, their effects could be addressed incrementally by adopting an appropriate research strategy during method development. A suitable strategy should support sequential advancement of the research as follows :

(i) iterative method development and *demonstration* of design support while indirectly ensuring field validity and the realisation of superior design artefacts;
(ii) field *validation* of developed methods.

Such a strategy would comprise the following :

(1) the recruitment of established human factors methods and knowledge. By building upon established research and design practice, it would be reasonable to expect that field validity will accrue to the structured method being developed. Similarly, it may be expected that superior design artefacts would result from its application;
(2) the adoption of stringent methodological requirements entailed by existing SADMs. Emulating the systematic and comprehensive design emphasis of SADMs ensures an orderly design process in the structured method being developed. Since an orderly design process encourages more complete problem analysis, it would be reasonable to expect that superior design artefacts would result from the application of the method.

The adoption of such a research strategy is additionally motivated by resource constraints, i.e. consideration (II) above. In particular, since method evaluation constitutes only part of the research concerns and more than one test cycle is desired, field tests should be deferred until the method is sufficiently well developed. Such tests would exact unacceptably heavy demands on project

resources since they would require extensive longitudinal and lateral studies to control for the following :

(a) designer experience and capability;
(b) characteristics peculiar to the particular system development project, e.g. economic pressures;
(c) organisational and social influences on the conduct of the design;
(d) design team composition.

Thus, by partially separating method development from method validation concerns, the strategy decomposes the research into more manageable modules. A suitable scope for method evaluation that is commensurate with available research resources may then be defined. Specifically, in the case of the method development module, method *demonstration* tests would predominate. Thus, the nature and scenario of method evaluation may be summarised as follows :

(i) nature of tests : *demonstration* before *validation* of method capability; *direct* address of design support capability and *indirect* address of the realisation of superior design artefacts;
(ii) evaluation scenario : demonstration of method capability by *method developers, 'in-house' designers* and *design teams* (in that order), before field validation with *other designers* and *design teams*.

It should also be noted that the above ordering of evaluation scenario is consistent with the procedure for methodological integration, i.e. derivation of a satisfactory structured human factors method before explicit integration with the chosen SADM. In other words, evaluation using a design team scenario is only applicable after an integrated method has been specified.

Two further concerns of method evaluation need to be addressed, namely the focus of evaluation and the type of test beds that should be selected.

Firstly, the evaluation could focus on functionality and usability characteristics of the method. Since the type of evaluation is influenced by the nature and scenario of

method evaluation (above), it may be inferred that functionality evaluation takes precedence over usability evaluation during method development. On deriving a satisfactory method, usability evaluation may then be conducted to assess the appropriateness of its expression. For similar reasons, subjective assessment takes precedence over objective assessment. Relevant subjective assessments of the method may include : completeness of design scope, flexibility, utility, acceptability, learnability, and compatibility with the SADM and other established design methods and practices, etc. In the longer term, sufficient data may be accumulated from its application to support objective assessments of the method. For instance, the method could be assessed in terms of its capability for facilitating improvements in design performance, e.g. quality of project management, design documentation and final design artefact; errors committed and detected during design development; project turn-around time; overall project resource requirements; etc.

Secondly, the above nature and scenario of method evaluation indicate that appropriate case-study systems should be selected as test beds during method development. To this end, the selection criteria comprise :

(a) the case-study system domain. During initial method development, the system domain should not be unnecessarily complex or unfamiliar to the method developer. Alternatively, the domain of the initial case-study system should not be ill defined. These criteria help to ensure maximum expenditure of research time and effort on method development as opposed to domain familiarisation. On deriving a reasonable method, other system domains may be considered to broaden the scope of method tests, e.g. to test the capability of its notation for describing different design domains;
(b) the size of the case-study system : for the same reason as in (a) above, the scale of case-study systems should be incremented progressively as the method is developed. Alternatively, successively larger modules of a case-study system may be selected as test beds;
(c) the complexity of the case-study design scenario : for the same reason as in (a) above, a simpler design scenario should be selected at early stages of method development. For instance, a *variant* design scenario would be

more suitable than a *novel* design scenario. Following the development of a satisfactory method, tests under a novel design scenario may be introduced, e.g. to test elicitation procedures of the method more fully.

The preceding account describes research concerns which apply generally to the structured integration of human factors methods with any SADM. The research concerns may be decomposed into more detailed activities (see Figure 3-1). A research plan comprising modular schemes and strategies, may then be specified to manage the activities more efficiently. A description of such a plan follows.

3.3. A General Research Plan for Methodologically Integrating Human Factors with SADMs

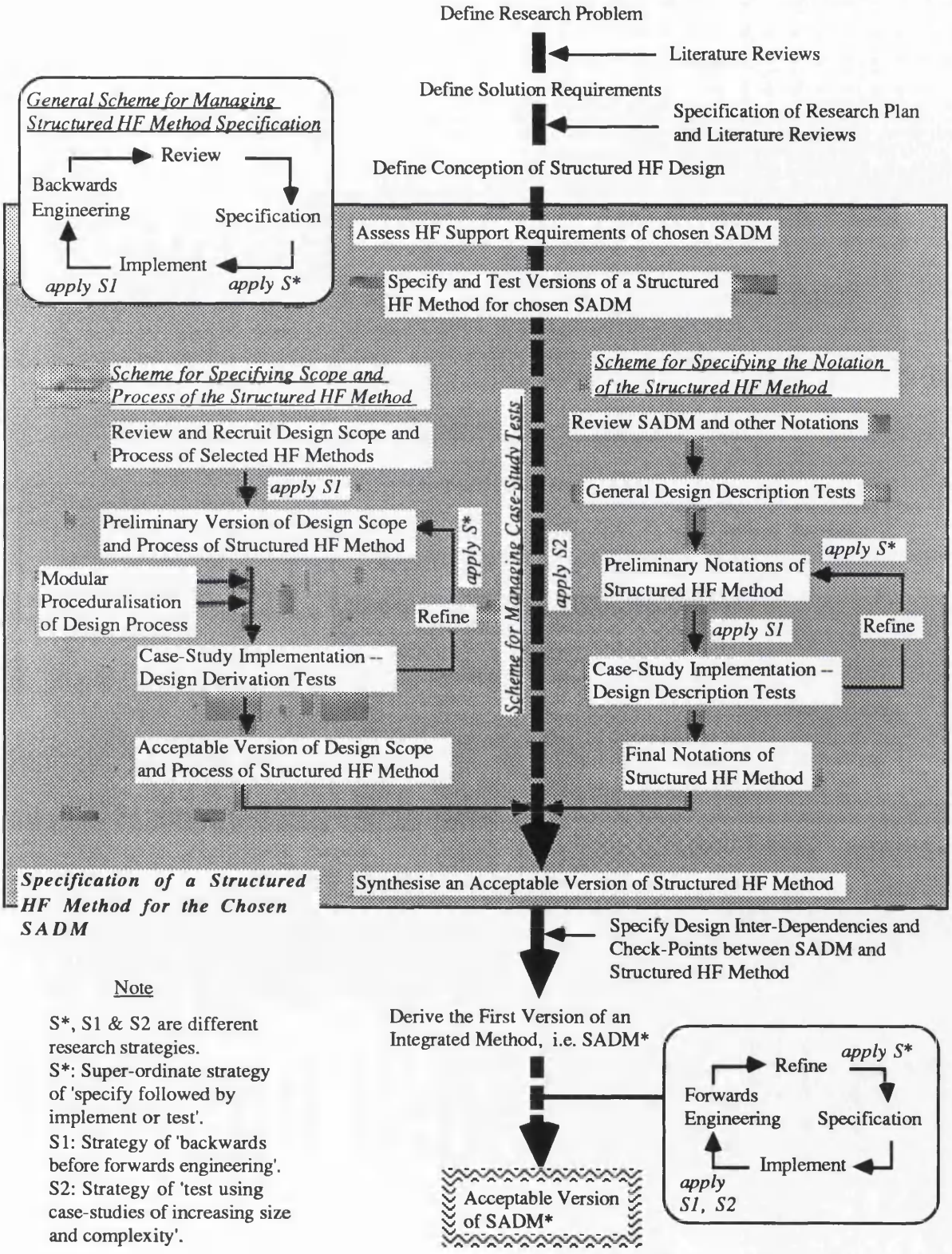
To manage the research effectively, an appropriate plan is necessary. The function of such a research plan is three-fold :

(a) to arbitrate between alternative ways of conducting the research;
(b) to support method specification, e.g. derivation of design procedures;
(c) to support the implementation of case-study tests and iterative method development.

Figure 3-1 shows a general research plan for the structured integration of human factors methods with SADMs. Note that the integrated method is referred to as a SADM*. As shown in the Figure, the research plan essentially comprises a schedule of activities operationalised by modular schemes and strategies. Specifically, the research is made more tractable by the application of three strategies, namely S*, S1 and S2. A review of the plan follows.

The plan is initiated by a problem statement which identifies general objectives of the research. In the present context, the objective is to facilitate the design of better human-computer systems by ensuring human factors input throughout the system design cycle (see Chapter One). Literature reviews may be conducted to detail the research problem further. For instance, the current problem of human factors input

Figure 3-1 : A Research Plan for Specifying an Integrated Method



Note

S*, S1 & S2 are different research strategies.
S*: Super-ordinate strategy of 'specify followed by implement or test'.
S1: Strategy of 'backwards before forwards engineering'.
S2: Strategy of 'test using case-studies of increasing size and complexity'.

was revealed as the 'too-little-too-late' problem. In addition, previous research solutions should be reviewed to uncover potential pitfalls and useful ideas.

New design support requirements and solutions may then be outlined. For instance, a requirement may be the explicit location of appropriate inter-disciplinary design concerns with respect to the system design cycle. Making design intersections between Human Factors and Software Engineering explicit ensures that human factors inputs are optimally relevant to design support needs at each stage of system development, i.e. ensuring the timeliness, format and granularity of human factors inputs. A solution that satisfies these requirements may be to integrate Human Factors and Software Engineering methods structurally. Since the design structure of SADMs is explicitly defined throughout the system design cycle, its recruitment would facilitate the structured integration of Human Factors with Software Engineering methods (see Chapter Two). Similarly, it follows that integration would also be facilitated by a correspondingly structured human factors design method. Since an adequately structured human factors method does not exist, it needs to be specified (see earlier sub-sections). To this end, existing conceptions of human factors design are surveyed and recruited as appropriate. The recruited conceptions are then extended and synthesised to derive a *structured* conception of human factors design (see Chapter Four). The structured conception constitutes the basis for specifying a structured human factors method. For instance, the scope of such a method may be identified by intersecting the human factors design conception with the support requirements of the chosen SADM. Thus, a compatible structured human factors method may be specified and integrated with the SADM.

To support the specification and integration of structured methods, previous reports of similar research should be reviewed (see Chapter Five). The review may indicate an appropriate SADM for human factors integration, e.g. a SADM that was previously unexplored.[5]

[5] The selection of the JSD method for the present research is dictated primarily by its sponsors (see Chapter Six).

Following the selection of a SADM, its requirements for human factors design support should be identified (see Chapter Six). Generally, the requirements may be extracted from two sources, namely by :

(a) reviewing previous reports of the inadequacies of the chosen SADM and SADMs in general;
(b) comparing the design scope addressed by the SADM against the structured conception of human factors design derived earlier.

On the basis of these requirements, a structured human factors method may be specified using two research schemes, namely a scheme for specifying its design scope and process; and another for specifying its notation. These schemes may be applied in parallel if the notations of the chosen SADM are sufficiently developed and promising for describing human factors design products.

Figure 3-1 shows that early versions of a structured human factors method may be backwards engineered[6] using simple case-study systems (see Chapter Seven). Such systems are characterised by well defined design start-points (requirements) and end-points (implemented design artefacts). Following the specification of a preliminary structured human factors method, incrementally larger case-study systems and forwards engineering[7] tests may be applied (in that order). Successive versions of a structured human factors method are thus tested and refined iteratively.

On deriving an adequate structured human factors method, explicit integration with the chosen SADM may be attempted. Design inter-dependencies and check-points

---

[6] This process will be explained later when research strategies are described. Suffice it to say at present that reverse or backwards engineering is defined as the process of deriving a set of design specifications from a *finished* system, i.e. it entails a *post-hoc* examination by person(s) *other than* the developer(s) (Rekoff, 1985). A design process for the system is thus revealed.

[7] Forwards engineering is the opposite of reverse or backwards engineering. It may be viewed as the 'normal' process of design specification.

are thus specified to co-ordinate design advancement between the two component streams of the integrated method (or SADM*). The SADM* is then subjected to further forwards engineering tests. The primary objective is to assess the appropriateness of the design inter-dependencies and check-points. Thus, successive versions of the integrated method are tested and refined as appropriate. The method refinement and test cycle is continued until a satisfactory SADM* is derived.

More needs to be said about the strategies that support the research. In particular, three research strategies are applicable, namely :

(a) a super-ordinate strategy (S*) that involves iterative cycles of method specification followed by its implementation in case-study tests. At the end of each cycle, the method is upgraded. Such a strategy is analogous to hypothesis testing, i.e. a method is specified (the hypothesis) and then implemented in the design (the test) of a case-study system (the test bed);
(b) a sub-ordinate strategy (S1) that facilitates the specification of early versions of a structured human factors method. The strategy involves backwards engineering before forwards engineering (see Figure 3-1). Backwards or reverse engineering is a technique for deriving intermediate design specifications and processes for a developed product. Although the technique originates from hardware design, it is increasingly applied in software design to rectify poorly managed projects, e.g. for post-implementation design recovery and re-documentation (Chikofsky and Cross II, 1990). Thus, backwards engineering could support the early stages for specifying a structured human factors method. Specifically, by applying the technique iteratively under various design scenarios for a class of design artefacts (e.g. software user interfaces), a generic design process may be abstracted for that class. The generic design process may then be subjected to forwards engineering tests to assess its capability for supporting the design of artefacts within that class. In the present context, backwards engineered versions of a structured method may be assessed initially by simulating a variant design (forwards engineering) of the same

case-study system. On deriving a satisfactory method, other case-study systems (or another module of the current case-study system) may be introduced as test beds.

During method tests, answers to the following questions are sought :

(i) are design processes and intermediate design products of the method sufficiently well defined to support human factors design, e.g. user interface specification (Figure 3-1, left side) ?
(ii) can notations of the method describe intermediate design products adequately to facilitate inter-designer and designer-user discussions ? Are they also specific enough to support unambiguous implementation of human factors specifications (Figure 3-1, right side)?
(iii) are the specified design inter-dependencies and check-points sufficient to ensure efficient design convergence between the component streams of the SADM* ? Are unnecessary design constraints imposed by these inter-dependencies and check-points (Figure 3-1, lower part) ?

Thus, appropriate upgrades to the current version of the method may be inferred.

(c) a second sub-ordinate strategy (S2) that supports the selection of appropriate case-study systems for testing the method. Specifically, strategy S2 prescribes the separation of test-bed complexity (e.g. familiarity with the domain of the case-study system) from method development concerns. For instance, strategy S2 could be applied to support strategies S1 and S* as follows :

(1) use backwards engineering to derive a preliminary version of the structured human factors method (strategy S1);
(2) test method iteratively (strategy S* -- see (3) below) under a forwards engineering scenario (strategy S1), using case-study systems of increasing complexity and size (strategy S2);

(3) iteratively develop the method via method specification and test cycles (strategy S*).

A variant of strategy S2, namely the separation of method development concerns, may also be applied to control the complexities of the research. Specifically, strategy S2 prescribes that during the initial stages of method development, the research should focus on the *definition* of stage-wise design products entailed by the method. On adequate definition, attention may then be directed on their *derivation.*

This account completes the description of a general research plan for the specification of an integrated method. In implementing the research plan, specific research constraints need to be addressed. An instantiation of the general research plan is presented as follows :

(a) Chapters Four and Five summarise the outputs of literature reviews prescribed by the research plan for initiating method development. Specifically, existing conceptions of human factors design were reviewed and recruited (as appropriate) to the specification of a structured conception of human factors design (Chapter Four). In addition, previous research on human factors integration with SADMs were reviewed to inform the present research on past successes and failures (Chapter Five). With minor exceptions, the outputs of these reviews are general to this class of research;
(b) Chapters Six and Seven describe an implementation of the schemes and strategies entailed by the general research plan. Detailed research activities for specifying a particular SADM*, namely an integrated Jackson System Development method (or JSD*), are thus operationalised. The content of these chapters are therefore specific to the present research. Their objective is to establish the context for subsequent discussions on the JSD* method.

Chapter Four : Towards a Conception of Structured Human Factors Design

"A precedent embalms a principle."
Lord Stowell, Advocate General, 1788.

In previous Chapters, the importance of earlier and wider human factors design involvement was highlighted. Consequently, additional areas of human factors inputs have been identified. However, the inputs mapped poorly onto the design support needed at various stages of the design cycle since the process of human factors design remains largely implicit. The unsatisfactory state of affairs results directly from the historically late recruitment of human factors in system design. Specifically, late recruitment predisposed its design methods to a narrow coverage of the system design cycle. Thus, to alleviate the observed problems an adequately structured conception of human factors design is a pre-requisite. In addition, a structured conception would facilitate the integration of human factors methods with SADMs. For instance, the requirements for structured integration may be defined by intersecting the scope of the conception with the chosen SADM. Relevant human factors methods may then be recruited to the specification of a *structured* human factors method that is appropriate for integration with the SADM (see Chapter Seven).

To this end, the derivation of a structured conception of human factors design constitutes the focus of this chapter. Thus, its objectives comprise the following :

(a) to review existing conceptions of human factors design and then collate them into a 'consensus' conception. On the basis of the 'consensus' conception, an initial conception of *structured* human factors design may be derived (see (b) below);
(b) to construct analytically a *structured* human factors design framework. The framework is derived by extending the 'consensus' conception via the application of basic human factors design premises. Thus, stage-wise manipulations of human factors design primitives (namely task, human,

device and environment) may be specified;

(c) to assess existing conceptions of *structured* human factors design and to collate them with the analytic framework derived in (b) above. An enhanced conception of structured human factors design is thus derived. The latter conception is then used to identify existing human factors techniques and methods that may be recruited to the construction of a structured human factors *method.*

These objectives are discussed in the sub-sections that follow.

4.1. A Survey of Existing Conceptions of Human Factors Design

To derive a 'consensus' conception of human factors design, a review of relevant reports by researchers and practitioners was conducted. The survey highlighted the following :

(a) the coverage of human factors design is confined to a narrow part of the system design cycle. Thus, it may be concluded that human factors design is generally incomplete;

(b) the taxonomy and scope of existing human factors design concerns are poorly defined. As such, the location of human factors concerns with respect to the system design cycle are inconsistent across reports. For instance, Shackel (1986a) located task analysis after functional design while Grudin et al (1987); Haubner (1990); Pikaar et al (1990); etc. reported the reverse order;

(c) there is no agreement on the scope of human factors input to system design. For instance, Mantei and Teorey (1988) included pre-design product acceptance analysis, and Haubner (1990) included post-implementation product surveys;

(d) very few reports include an explicit identification of the inputs required for human factors design and the intermediate design products that are derived subsequently. Reports that fulfil this criteria were generally dedicated to design analysis at specific stages of the design cycle, e.g. task

analysis. Thus, it may be concluded that human factors design is generally implicit;

(e) very few reports identify explicit relationships between human factors design stages and describe how design should be advanced. For instance, it is unclear what outputs of a particular design stage would constitute inputs to a succeeding design stage. Of the hundred or more reports surveyed, only two addressed this issue, namely those by Shackel (1986a) and Jones (1973). Other reports generally described human factors design as comprising a vague ordering of design concerns. Such reports would only be informative with respect to *what* human factors concerns should be considered but not *how* they should be addressed during design;

(f) there are two levels of human factors input to system design, namely at the organisational level and the interactive work system level. These levels may correspond to system and sub-system design respectively.

A summary of reports selected from the above human factors design survey is shown in Tables 4-1 to 4-4. It can be seen from the Tables that human factors design was described as a mixture of design process and product concerns. Also, the design concerns were neither described at the same level nor mutually exclusive. For instance, it is unclear across reports whether task elicitation and description should be included in task analysis (these were considered distinct concerns in Shackel (1986a)). For these reasons, a 'consensus' conception of human factors design can not be inferred directly from these reports. To derive such a conception, a simple ranking scheme had to be applied. The procedures of the scheme are as follows :

(a) the number of items in Tables 4-1 to 4-4 was reduced by grouping similar items under one category, e.g. user analysis and user characterisation are grouped under the latter. Similarly, subsets are grouped into a super-ordinate set, e.g. task analysis and task description are categorised under the former. Minor or 'non-main stream' human factors concerns are excluded, e.g. late customisation and product survey. To reduce the categories of human factors concerns further, a 'basic' set was selected from the reports by imposing an acceptance threshold. Specifically,

only design concerns that were mentioned in more than fifty percent of the reports would be selected for inclusion in the basic set.[8] Thus, nine categories of human factors concerns were identified. Following identification, human factors concerns cited in each report were classified under the categories (Tables 4-1 to 4-4). The results are shown in Table 4-5; (b) the reported sequence of human factors design categories was noted and ranked for each of the reports (refer to Tables 4-1 to 4-4 -- pull the tabs for a complete view). As the number of categories across the reports are different, the rankings (raw score) were normalised to the maximum number of categories,[9] i.e. to base nine. The modal and adjacent score(s) of the normalised set (shown in parenthesis in Table 4-6) were then considered to decide the sequence of the nine categories of human factors design. The sequence thus constitutes a 'consensus' view of human factors design for the surveyed reports.

The derived 'consensus' conception of human factors design may be summarised as follows :

System level considerations

Stage 1 : System performance definition (i.e. requirements specification)

(Stage 1a : User characterisation)

Stage 2 : Function allocation

(Stage 2a : User characterisation, Extant system task analysis, task synthesis, Environmental design & Training projection)

---

[8] Relevant contextual information should be noted to support more specific insights into human factors design. For instance, appropriate contextual information would support a lower level interpretation of the 'consensus' conception, e.g. to uncover why task analysis is located at various design stages.

[9] A sample calculation for normalising a score from base 8 to base 9 (see Table 4-6, column 2, row 2, Eason (1987)) is as follows : 2 (raw score) x 9/8 = 2 (normalised score to nearest integer).

Table 4-1 : Reported Human Factors Concerns and their Approximate Sequencing during System Design I

Design Phase	Eason (1987)	Mantei and Teorey (1988)	Pikaar et al (1990)
Feasibility	-- Analysis of socio-technical technical options -- Cost-benefit assessments of organisational impact	-- Product acceptance analysis	-- Orientation to system purpose
	-- Organisational and task analysis -- Job design and organisational change	-- Task analysis	-- System description and task analysis -- Global and detailed task allocation -- Job design and work organisation
	-- User specification of technical system -- Usability design -- Prototype evaluation	-- Global design -- Prototype evaluation	-- Workplace and man-machine interface
Implementation and evaluation	-- Customisation -- Workstation and environmental design -- User support -- System evolution	-- User testing -- Product survey	-- User testing

Table 4-1 : Reported Human Factors Concerns and their Approximate Sequencing during System Design I

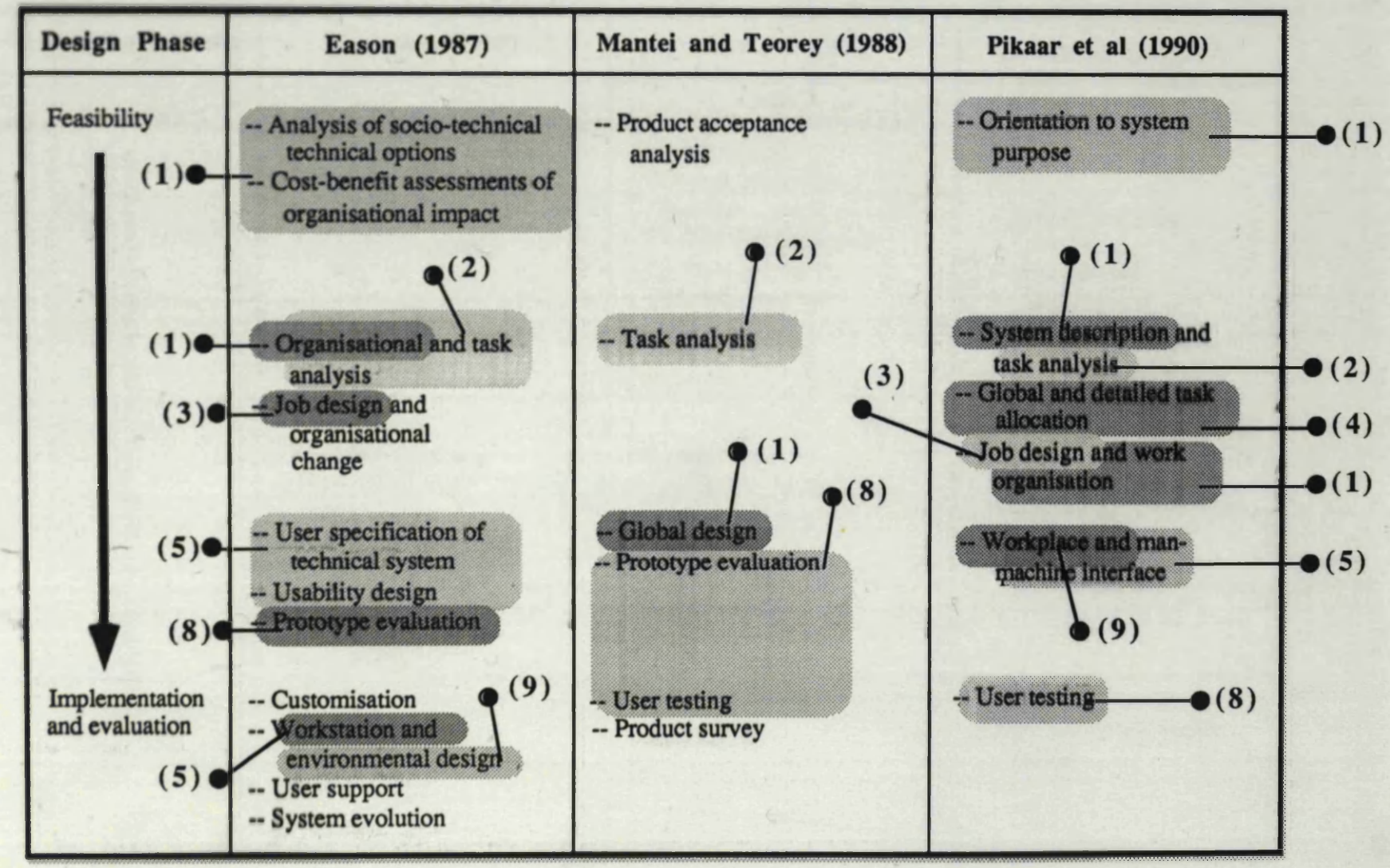


Table 4-2 : Reported Human Factors Concerns and their Approximate Sequencing during System Design II

Design Phase	Haubner (1990)	Shackel (1986)	Grudin et al (1987)
Feasibility	-- Task and user group analysis	-- Define system purpose and performance -- Compare with other systems -- Identify user population -- Function allocation -- Task synthesis and description -- Job specification -- User selection criteria	-- Needs elicitation -- User focus groups -- Product mock-up -- User testing -- Detailed task analysis -- Functional design
	-- Function allocation -- Task and system description -- User interface design -- Information coding -- Dialogue and input design	-- Task analysis -- Workstation design -- Job aids design -- Organisation and management design -- Training design	-- User testing and iterate design
Implementation and evaluation	-- User testing -- Product survey	-- User testing	-- User testing and iterate design

Table 4-2 : Reported Human Factors Concerns and their Approximate Sequencing during System Design II

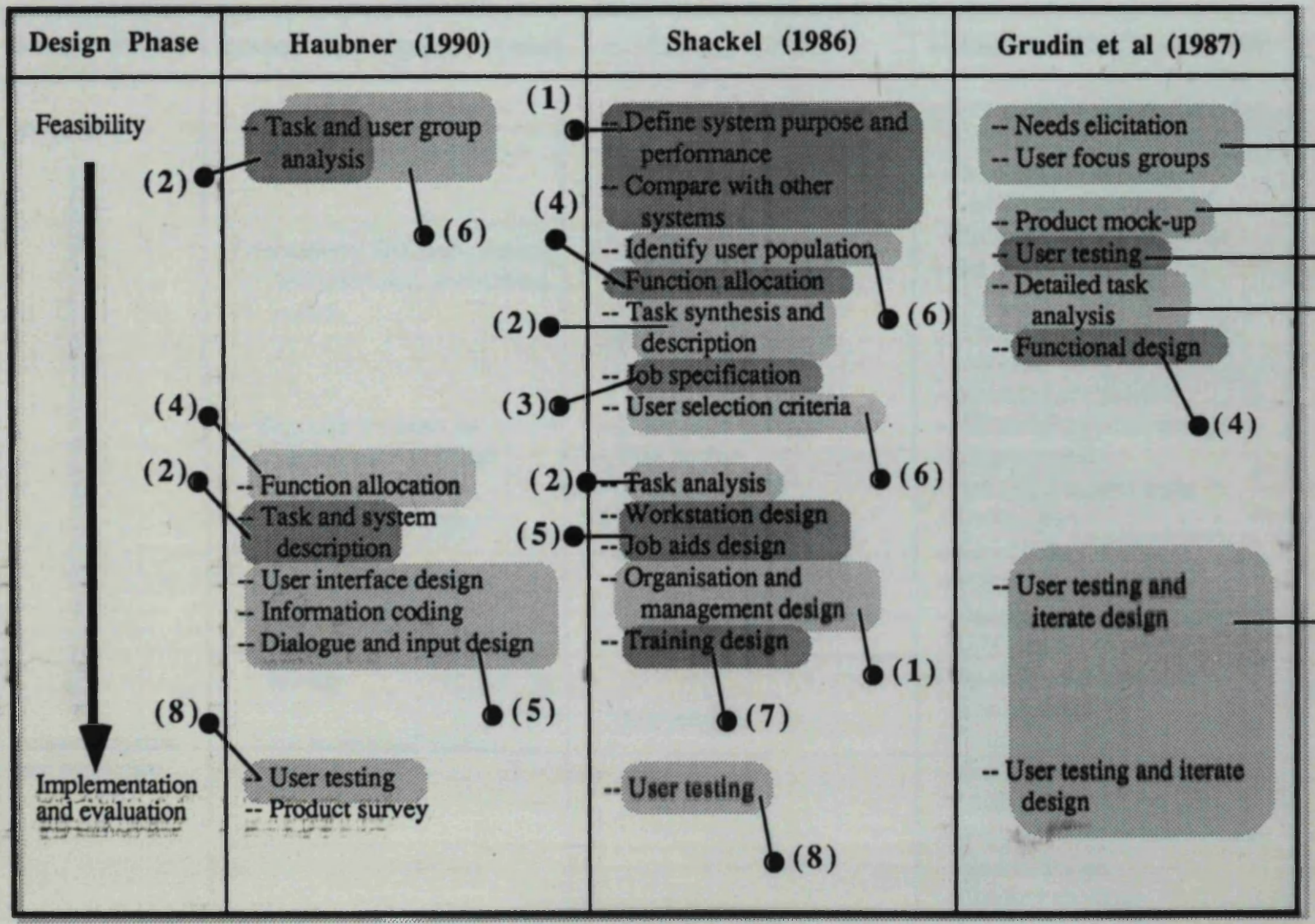


Table 4-3 : Reported Human Factors Concerns and their Approximate Sequencing during System Design III

Design Phase	Kloster and Tischer (1984)	Clowes (1986)	Gillett and Northam (1990)
Feasibility			-- Characterise skill profile of users -- Determine operations environment -- Estimate manpower needs -- Set performance and effectiveness criteria -- Analyse and design overall system
	-- Functional and performance definition of man-machine system	-- User analysis and characterisation -- Task analysis -- Environment analysis	-- Specify sub-systems -- Allocation of task functions -- Task analysis -- Set HCI dialogue style -- Design jobs -- Set training criteria -- Prototype sub-systems -- Specify operation procedures -- Evaluate sub-systems -- Specify team operations -- Specify roles -- Training design -- Evaluate overall system -- Validate training programme -- Job performance assessment -- Update design -- Conduct training
	-- Organisation analysis -- Training and workload assessment -- Environment design -- Job design -- Task analysis -- Function allocation -- Dialogue and workstation design	-- Conceptual design -- Task design -- Semantic design -- Training design -- Dialogue design	
Implementation and evaluation	-- User testing and system evolution	-- User testing	

Table 4-3 : Reported Human Factors Concerns and their Approximate Sequencing during System Design III

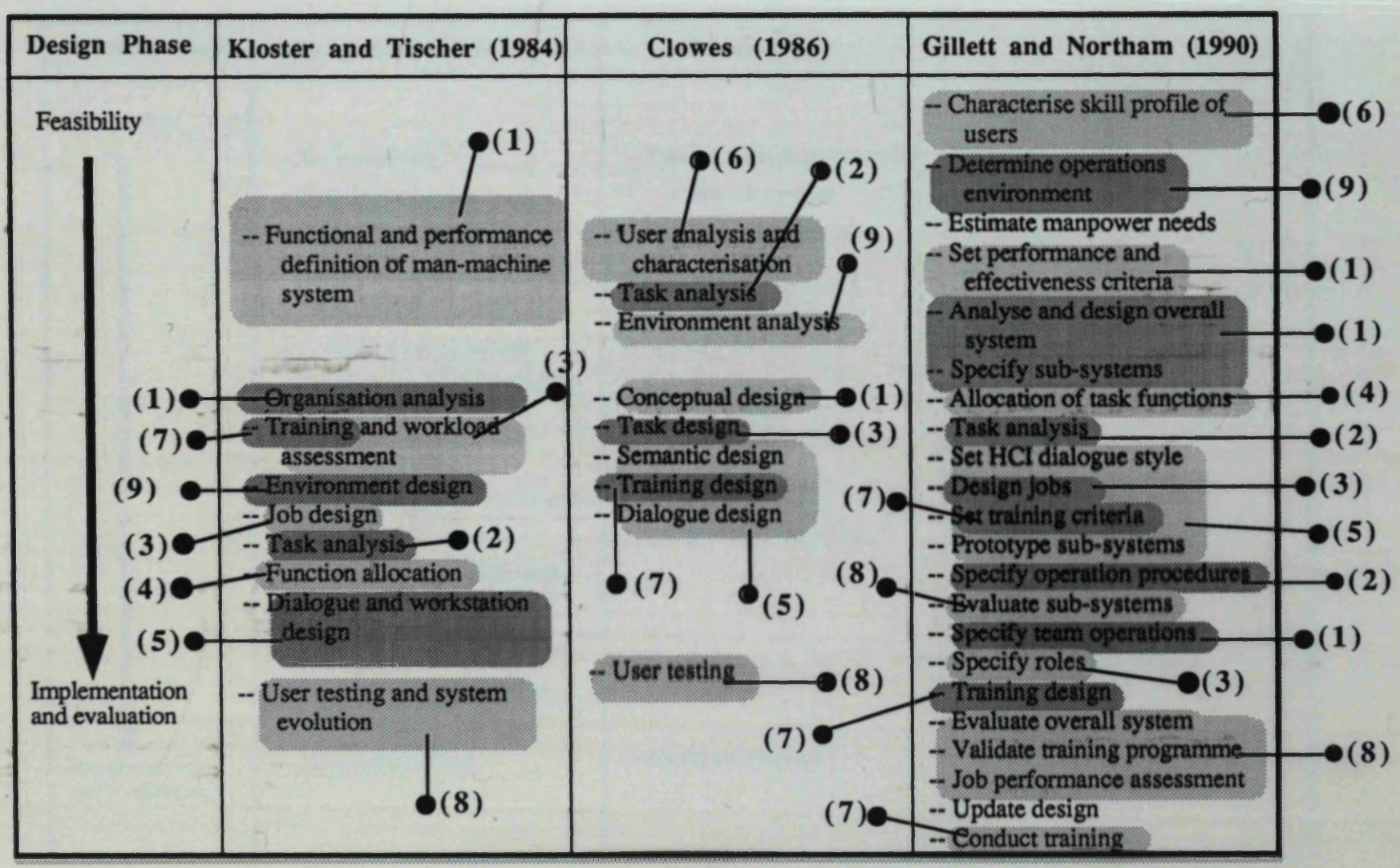


Table 4-4 : Reported Human Factors Concerns and their Approximate Sequencing during System Design IV

Design Phase	Pelly and Crampin (1990)	Jones (1973)	
Feasibility ↓ Implementation and evaluation	-- Task analysis -- Training analysis -- Function allocation -- User characterisation -- Task design -- Specification of human performance and in-built tests -- Training design -- Workstation and workplace design -- Environmental health and safety -- Job design -- Team work organisation -- MMI design -- Prototype testing	-- Specify inputs and outputs of overall system -- Specify functions of sub-systems -- Function allocation -- Specify human performance -- Design MMI -- Design job aids -- Design training -- Integrate sub-systems -- Evaluate system	

Table 4-4 : Reported Human Factors Concerns and their Approximate Sequencing during System Design IV

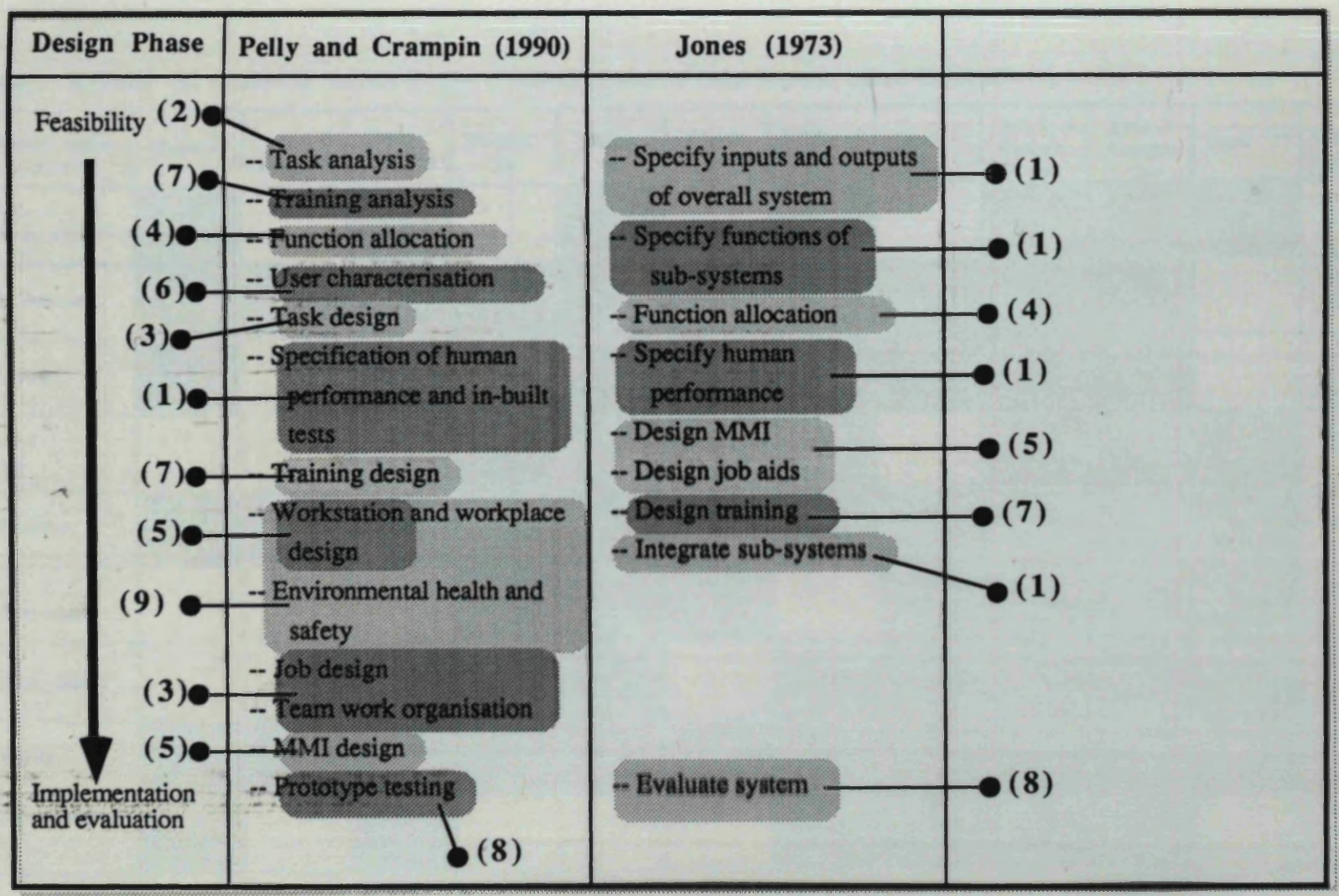


Table 4-5 : A 'Basic' Set of Human Factors Design Concerns Extracted from Reports Listed in Tables 4-1 to 4-4

Human Factors system design concerns	Eason'87	Mantei & Teorey '88	Pikaar et al '90	Haubner '90	Shackel '86	Grudin et al '87	Kloster & Tischer '84	Clowes '86	Gillett & Northam '90	Pelly & Crampin '90	Jones '73	*Number of times cited*
(1) System performance definition	■		■		■		■		■	■	■	7/11
(2) Task analysis	■	■	■	■	■	■	■	■	■	■		10/11
(3) Job design	■		■		■	■	■	■	■	■		7/11
(4) Function allocation			■	■	■	■	■		■	■	■	8/11
(5) MMI design	■	■	■	■	■		■	■	■	■	■	9/11
(6) User characterisation				■	■	■		■	■	■	■	7/11
(7) Training design					■		■	■	■	■	■	6/11
(8) Evaluation	■	■	■	■	■	■	■	■	■	■	■	11/11
(9) Environment design	■		■				■	■	■	■		6/11

Table 4-6 : Deriving the Relative Sequence of Human Factors Design

Human Factors system design concerns	Eason'87	Mantei and Teorey '88	Pikaar et al '90	Haubner '90	Shackel '86	Grudin et al '87	Kloster and Tischer '84	Clowes '86	Gillett and Northam '90	Pelly and Crampin '90	Jones '73	*Modal Score*	*Adjacent Score(s)*
(1) System performance definition	1, 3 (1, 3)	2 (6)	1, 5 (1, 6)	---	1, 9 (1, 7)	---	1 (1)	4 (4)	3, 12 (2, 7)	6 (5)	1, 3, 6 (1, 4, 8)	1	---
(2) Task analysis	2 (2)	1 (3)	2 (2)	1, 4 (2, 6)	4, 7 (3, 6)	4 (6)	6 (6)	2 (2)	5, 10 (3, 6)	1 (1)	---	6	2
(3) Job design	4 (5)	---	4 (5)	---	5 (4)	---	3, 5 (3, 5)	5 (5)	7, 13 (4, 7)	5, 10 (4, 8)	---	5	---
(4) Function allocation	---	---	3 (3)	3 (5)	3 (2)	5 (8)	7 (7)	---	4 (2)	3 (2)	2 (3)	2	3
(5) MMI design	5, 7 (6, 8)	---	7 (8)	5 (8)	8 (7)	2 (3)	8 (8)	6, 8 (6, 8)	6, 9 (3, 5)	8, 11 (6, 8)	4 (5)	8	6
(6) User characterisation	---	---	---	2 (3)	2, 6 (2, 5)	1 (2)	---	1 (1)	1 (1)	4 (3)	---	1, 2, 3	---
(7) Training design	---	---	---	---	10 (8)	---	2 (2)	7 (7)	8, 14, 16 (5, 8, 9)	2, 7 (2, 5)	5 (6)	8	---
(8) Evaluation	6 (7)	3 (9)	8 (9)	6 (9)	11 (9)	3, 6 (5, 9)	9 (9)	9 (9)	11, 15 (6, 8)	12 (9)	7 (9)	9	---
(9) Environment design	8 (9)	---	6 (7)	---	---	---	4 (4)	3 (3)	2 (1)	9 (7)	---	7	2

Numbers enclosed in parenthesis indicate the positional score normalised to base 9. The normalised score is then used to determine the relative scores in the last two columns. A sample calculation of the normalised score is as follows : for the Eason '87 column, the 'System performance definition' (Category (1)) was assigned a raw score of 1 out of 8; the normalised score (to base 9) is then 1 x 9/8 = 1.1 or 1 (rounded to nearest integer).

Stage 3 : User characterisation
(Stage 3a : Function allocation)
Stage 4 : Job design
(Stage 4a : Training design)
Stage 5 : Task analysis
(Stage 5a : User interface design)
Stage 6 : Environmental design
Stage 7 : User interface design & Training design
Stage 8 : Evaluation

(The italics indicate human factors concerns that may also be relevant at the design stage.)

Although the 'consensus' conception seems reasonably complete with respect to the system design cycle,[10] the underlying human factors design assumptions and 'logic' remain implicit. To support a better understanding of human factors design, its design variables and manipulations should be made explicit. One solution is to extend the 'consensus' conception into a structured design framework. The framework may be derived analytically by extending the conception with respect to basic primitives of human factors design, namely the task, user, environment and device. The derivation of such a framework is described in the next sub-section.

[10] The completeness of a conception is a function of the state of human factors knowledge at a particular point in time. It may be determined directly by conducting exhaustive tests on promising conceptions. Alternatively, it may be ensured implicitly by deriving a 'consensus' conception via a sufficiently wide review of reported conceptions. For completeness, the derived conception should be extended analytically. Finally, its completeness with respect to the system design cycle is assessed against the scope of existing Software Engineering SADMs.

4.2. Analytic Derivation of a Simplified Framework for Structured Human Factors Design

The preceding sub-section highlighted that the 'consensus' conception of human factors design is inadequately explicit and structured. Thus, it was suggested that basic premises of human factors design (comprising its design approach, primitives and assumptions) should be examined analytically to derive a structured human factors design framework. The framework may then be imposed on the 'consensus' conception to derive a structured conception of human factors design.

Generally, human factors design may be viewed as an extension of Grandjean's (1988) concept of user-centered design. Thus, the objective of the design is to : *'fit <x> to the human'* where x = task + device + environment. Specifically, the perspective implies the following system[11] design stages :

(a) identify the user population;
(b) define performance requirements[12] of the human-machine system;
(c) define requirements of the user population with respect to the task, environment and device;
(d) user-centered specification of the conceptual task, interactive task, environment, user interface and workstation.

Any 'shortfall' in the specifications of (d) that can not be rectified to meet the requirements in both (b) and (c) (e.g. owing to technological limitations), may then be 'compensated' by appropriate personnel training and selection. In other words, *'fitting the human to <x>'* should follow *'fitting <x> to the human'*.

---

[11] A system may comprise one or more sub-systems. Systems generally comprise users and devices operating in a particular environment. They perform work by executing tasks to effect desired state changes of real world objects (Dowell and Long, 1989).

[12] Satisfactory system performance may be interpreted as the achievement of work requirements at an acceptable level of human and computer costs (Dowell and Long, 1989).

Similarly, in the context of human-computer systems, the objective of human factors design is to achieve system goals at acceptable user costs via appropriate configurations of the interactive task, device (both hardware and software) and environment. Thus, system design involves manipulating the attributes of these design primitives so that the desired system performance is met, i.e. :

Desired system performance = f {environment, task, device, user}

or $P_{system} = f\{E, T, D, U\}$

An expansion of design manipulations of these attributes follows.

Generally, system design is instigated by desired changes in system requirements. In turn, the changes may be attributed to required :

(a) improvements in current system performance from some value P_{system} to P'_{system} ;

(b) modifications of system or sub-system(s) design attributes at the same level of performance.[13]

Specifically, system requirements are addressed (e.g. pre-set training needs and environmental limits) and expressed in terms of necessary modifications and extensions of the current system (note that the current system may be a manual system). For instance, sub-systems may be defined, functions may be allocated between human and device components of the sub-systems, and socio-technical interactions[14] among sub-systems may be described explicitly in terms of their

---

[13] Note that the design activities in (a) and (b) above are not mutually exclusive. For instance, system performance improvements in (a) may necessitate sub-system design changes in (b), since the former is a function of the performances of the latter, i.e. :

$P_{system} = f [P_{sub\text{-}system1}, P_{sub\text{-}system2}, \cdots\cdots]$

[14] Since informal work relationships are extremely varied, they should be addressed as they are met.

work relationships and information exchanges. Thus, the purpose and performance of the target system is defined conceptually. The conceptual design is then detailed progressively by iterations of sub-system(s) design followed by their integration. Such a view of system design was reported by Eason (1987); Gillett and Northam (1990); Shackel (1986a); Jones (1973); and Pikaar et al (1990).

In accordance with the central tenet of human factors design (namely user-oriented design), sub-system design is initiated by a specific definition of users and their needs[15] (Grudin et al, 1987; Clowes, 1986; Gillet and Northam, 1990; Haubner, 1990; etc. -- see Table 4-1 to 4-4). Prominent characteristics of users are assumed and documented as sub-system design constraints.[16] On the basis of these constraints, attributes of other human factors design primitives (namely environment, task and device attributes) are then manipulated iteratively to meet sub-system performance requirements. Since these design primitives are interacting, design iterations should be assumed to pervade the overview that follows. In addition, the designer may apply the following design prioritisation strategies :

(a) firstly, determine the attributes of central, critical and limiting design primitives. In this way, constraints which increase with design advancement are accommodated by primitives that are more amenable to design manipulations. For instance, users' jobs and tasks are addressed before environment and device design. Thus, the latter is designed to accommodate the constraints imposed by the former;

(b) secondly, determine the attributes of *independent* and easily controlled design primitives. A clearer and more stable set of design constraints is thus defined for subsequent accommodation by *dependent* design primitives. For instance, environment design should precede user interface design. In

[15] User characterisation may include an assessment of the implications of changes to the current task, e.g. transfer of learning effects implicated by a particular target sub-system design.

[16] It can not be over-emphasised that the assumption does not imply the obviation of user testing.

this way, interactions among design primitives may be managed more effectively.

Bearing in mind preceding design decisions, the *task* to be performed by a particular user group is defined. Task design involves decomposing sub-system functions (comprising the human and device) so that on-line and off-line task components[17] are detailed sufficiently to support job and training design considerations. For instance, on-line tasks may be decomposed into interactive tasks. At this design stage, participatory design and task analysis techniques may be recruited. The preceding task design manipulations may be represented informally as follows :

$U \text{ -----> } U$ *(i.e. presumed user attributes are upheld)*

$T \text{ -----> } T'$ *(i.e. the existing task is re-designed)*

Since $T = T_{on\text{-}line} + T_{off\text{-}line}$:

$$T_{on\text{-}line} + T_{off\text{-}line} \text{ ----> } T'_{on\text{-}line} + T'_{off\text{-}line}$$

(i.e. new on-line and off-line tasks are defined)

$$T'_{on\text{-}line} = T'_{device} + T'_{interaction}$$

(i.e. new device tasks (T'_{device}) and interactive tasks ($T'_{interaction}$) are detailed).

Presently, system level descriptions of the social and physical *environment* may be specified at a lower level of description. Specifically, earlier socio-technical assumptions are made explicit and various design options for the physical environment may be investigated. In particular, :

(a) the macro environment may be tempered to a range that is acceptable to the target system, e.g. by air conditioning;

(b) a micro-environment may be created via an ancillary device, e.g.

[17] On-line and off-line tasks are device-supported and manual tasks respectively.

protective clothing to shield the user from adverse conditions;

(c) the user may be trained physiologically to tolerate periods of work in the environment, while the machine is designed to withstand the conditions, e.g. training fighter pilots to tolerate gravitational forces;

(d) a combination of all of the above.

A design option is selected following appropriate consideration of both the psychological and physiological implications of the stressor, and technological constraints on the solution. The design manipulations described thus far may be represented informally as follows :

$U \text{ -----> } U$) attributes of these primitives are

$T \text{ -----> } T'_{\text{on-line}} + T'_{\text{off-line}}$) carried forward to constrain the

$T'_{\text{on-line}} = T'_{\text{device}} + T'_{\text{interaction}}$) design of the environment (E).

$E \text{ -----> } \{E', E\}$ *(i.e. the existing environment (E) is either left unchanged, or some changes may be planned (E')).*

At this juncture, *device* design (comprising software, hardware and workstation design) may be undertaken. To this end, functional design is pursued via iterative decompositions of device and interactive task components of the on-line task, i.e. T'_{device} and $T'_{\text{interaction}}$ respectively. The resulting lower level descriptions should be consistent with design decisions and constraints that have been carried forward to this stage. In addition, the inputs and outputs of T'_{device} and $T'_{\text{interaction}}$ should complement each other. An appropriate user interface design may then be specified on the basis of $T'_{\text{interaction}}$ (e.g. its inputs and outputs)[18] and an organisation's in-house style (if any). These human factors design manipulations may be summarised

[18] Since all design decompositions up to and including the interactive task and user interface design are shaped by the adopted user model, a closer match between the designer's and user's model of the system may be expected. Appropriate device usability and functionality would thus accrue.

informally as shown below :

U -----> U) attributes of these primitives are
E -----> $\{E', E\}$) carried forward to constrain the
T -----> $T'_{on\text{-}line} + T'_{off\text{-}line}$) design of the device (D).
$T'_{on\text{-}line} = T'_{device} + T'_{interaction}$)

$T'_{interaction}$ ----> {sub-tasks, procedures, object-action pairs}
(i.e. the interactive task is decomposed to the device level)
T'_{device} ----> {device programs}
D -----> D'
(i.e. a functionally new device is designed to replace the old)
$\{T_{on\text{-}line}, D\}$ -----> $\{T'_{on\text{-}line}, D'\}$ or $\{T'_{device} + T'_{interaction}, D'\}$
(i.e. a new set of tasks and devices is derived)

A brief summary of other design scenarios as follows.

(i) Re-designing the user interface of a device

In this case, the off-line and device tasks are unchanged. Thus, the design manipulations may be summarised informally as follows :

$\{T_{on\text{-}line}, D\}$ -----> $\{T'_{on\text{-}line}, D'\}$ or $\{T_{device} + T'_{interaction}, D'\}$
(i.e. the device task is unchanged while interactive task characteristics are modified)
$T'_{interaction}$ ----> {sub-tasks, procedures, object-action pairs}
(i.e. the alternative interactive task is described at the device level)
T_{device} ----> {device programs}
(i.e. unchanged core application)
D -----> D'
(i.e. a new user interface is designed to replace the old)

(ii) Re-designing the off-line task without changing the on-line task (the device is thus unchanged).

Such a design scenario may be undesirable since it could imply an increased workload corresponding to increases in the off-line task. Alternatively, the user may be compelled to undertake further off-line tasks to compensate for inadequate on-line support. In both cases, user costs are increased to meet desired performance requirements, while device costs remain unchanged (since additional on-line support is not provided). The design manipulations may be summarised informally as follows :

T -----> T'

$T' = T_{on\text{-}line} + T'_{off\text{-}line}$

$\{T, D\}$ -----> $\{T', D\}$ or $\{T_{on\text{-}line} + T'_{off\text{-}line}, D\}$

(i.e. on-line task (and hence device design) is unchanged while off-line task is modified)

(iii) Re-designing manual tasks

In this case, on-line tasks do not exist. Thus, the design manipulations are summarised informally as follows :

$T = T_{on\text{-}line} + T_{off\text{-}line}$

$T = T_{off\text{-}line} = T_{manual}$ since $T_{on\text{-}line} = \{\}$ and $D = \{\}$

T -----> T'

===> T_{manual} -----> T'_{manual}

(iv) Automation of an existing set of manual tasks

The introduction of a device to support existing tasks essentially involves function allocation and the specification of on-line tasks. Thus, the design

manipulations may be summarised informally as follows :

$T = T_{on\text{-}line} + T_{off\text{-}line}$

$T = T_{off\text{-}line} = T_{manual}$ since $T_{on\text{-}line} = \{\}$ and $D = \{\}$

if D -----> D' and $D' \neq \{\}$

then T_{manual} -----> $T'_{on\text{-}line} + T'_{off\text{-}line}$ and $T'_{on\text{-}line} \neq \{\}$

(i.e. function allocation)

$T'_{on\text{-}line} = T'_{device} + T'_{interaction}$

$T'_{interaction}$ ----> {sub-tasks, procedures, object-action pairs}

T'_{device} ----> {device programs}

$\{T_{manual}, \emptyset\}$ -----> $\{T'_{off\text{-}line} + T'_{device} + T'_{interaction}, D'\}$

(i.e. the manual task is replaced by a new task comprising off-line, interactive and device tasks)

(v) Automating and extending an existing set of manual tasks

This scenario is a variant of (iv) above since the existing task is extended in addition to the introduction of a device. Thus, function allocation and on-line tasks have to be specified (not shown below -- see (iv) for further details). The design manipulations may be summarised informally as follows :

$T = T_{on\text{-}line} + T_{off\text{-}line}$

$T = T_{off\text{-}line} = T_{manual}$ since $T_{on\text{-}line} = \{\}$ and $D = \{\}$

if $T_{off\text{-}line}$ -----> $T'_{off\text{-}line}$ or T'_{manual}

and if D -----> D' and $D' \neq \{\}$

then T'_{manual} -----> $T''_{on\text{-}line} + T''_{off\text{-}line}$

$\{T_{manual}, \emptyset\}$ -----> $\{T''_{on\text{-}line} + T''_{off\text{-}line}, D'\}$

As a final consideration, *user* selection and appropriate training may be considered if the desired level of sub-system performance cannot be achieved through further design iterations. The design manipulations involved may be summarised informally

as follows :

T -----> $T'_{on\text{-}line} + T'_{off\text{-}line}$) attributes of these

$T'_{on\text{-}line} = T'_{device} + T'_{interaction}$) primitives are

E -----> $\{E', E\}$) carried forward to

$T'_{interaction}$ ----> {sub-tasks, procedures, object-action pairs}) constrain training

T'_{device} ----> {device programs}) design and user

D -----> D') selection criteria.

U -----> U'

(i.e. modifying user attributes by training and selection)

Although the preceding account is a simplified description of structured human factors design (e.g. stage-wise design iterations and evaluation have been omitted), its emphasis on stage-wise design manipulations supports a more specific interpretation of the 'consensus' conception of human factors design (see sub-section 4.1). Specifically, the structured framework supports the :

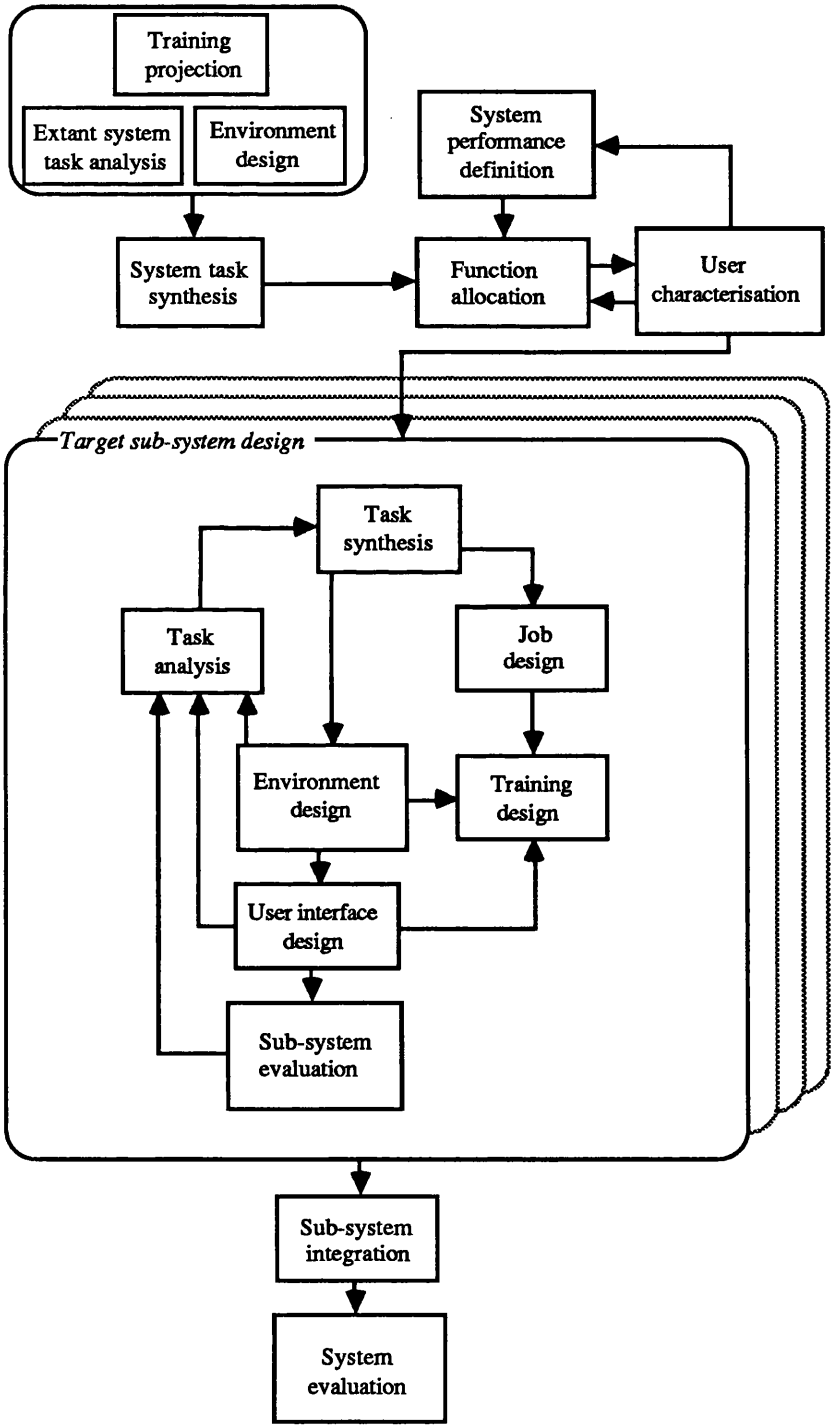
(a) inclusion of task synthesis at different levels of description. Specifically, task synthesis at a high level would precede functional allocation as the latter cannot begin without some notion of the system task. At a lower level, sub-system task synthesis supports human-machine interaction design (see Figure 4-1). Similarly, task description is subsumed in task synthesis and task analysis.

(b) interpretation of design relationships described by the 'consensus' conception. In addition, it supports the instantiation of these relationships in the initial conception of structured human factors design that is derived subsequently (see Figure 4-1).

The conception is then enhanced by incorporating further contributions from other structured design conceptions. On the basis of the enhanced conception, a structured human factors method may then be constructed. These research concerns

are addressed in the next sub-section.

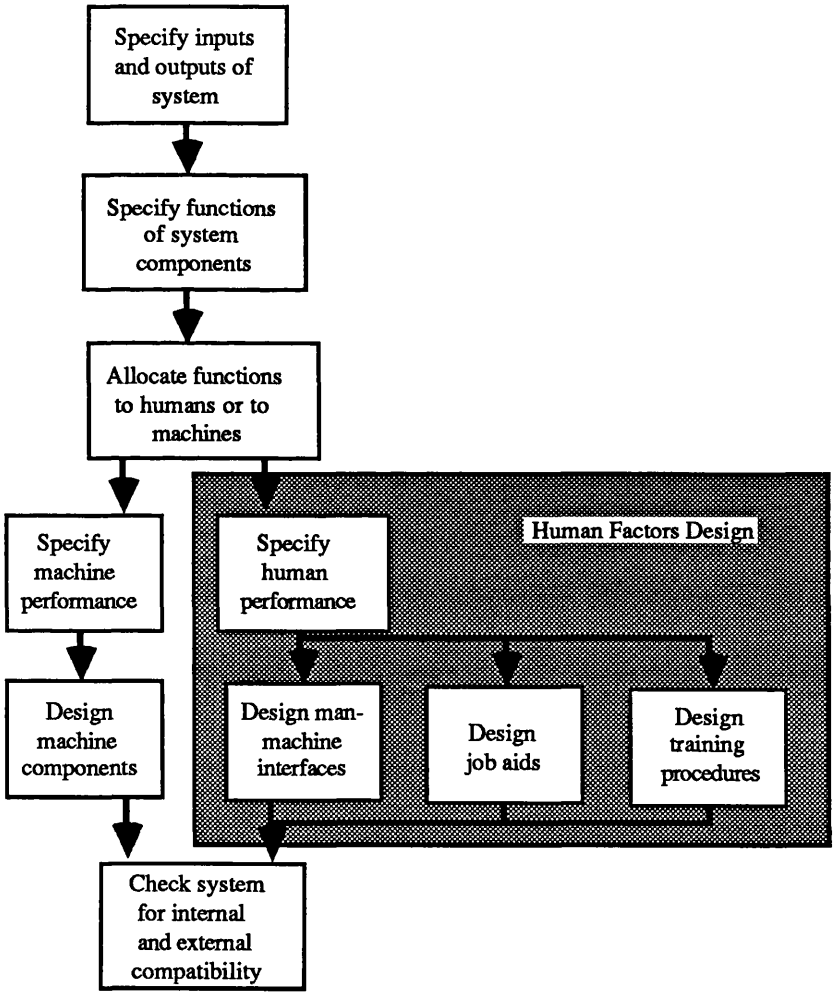
Figure 4-1 : An Initial Conception of Structured Human Factors Design



4.3. Towards an Enhanced Conception of Structured Human Factors Design

In 1973, Jones proposed a general conception for designing human-machine systems (Figure 4-2) that accommodates early ideas on function allocation, e.g. as suggested by Fitts (1962), Chapanis (1965) and Singleton (1972).

Figure 4-2 : A Conception of the Structured Design of Human-Machine Systems (Jones, 1973)



Although the conception is rather simple, it highlights the following human factors

design concerns :

(a) specification of user requirements (expressed as human performance setting) *prior to* user interface design;
(b) separation of human and machine design;
(c) design of user interfaces *in parallel* with ancillary support and training.

Since then, the scope of human factors design has grown to include :

(a) conceptual definition of the target system, e.g. task and information flows of the system;
(b) functional specification, e.g. human-machine allocation;
(c) organisational and socio-technical design;
(d) job specification and personnel selection;
(e) system integration and evaluation.

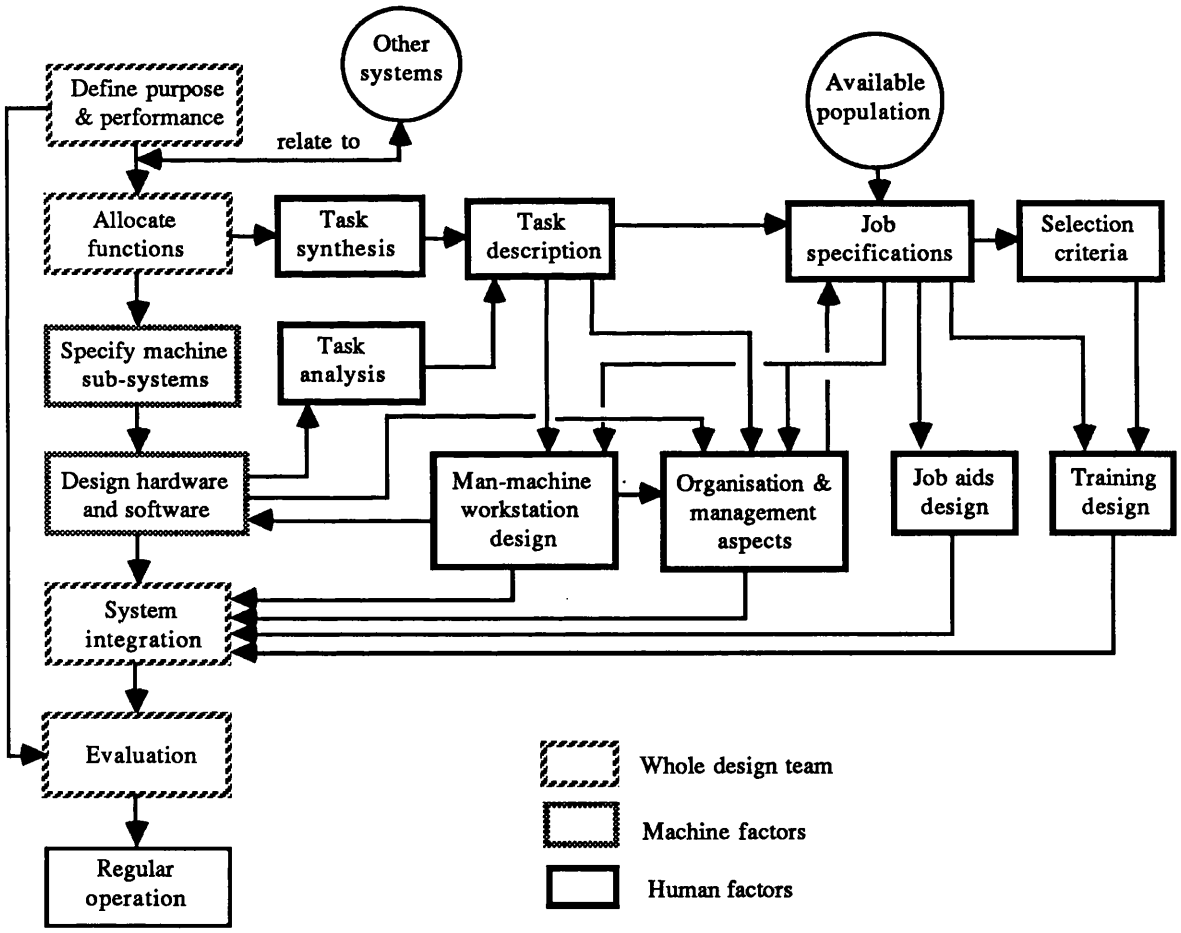
Thus, the scope of Jones' conception was updated appropriately (see Shackel, 1986a). Although the updated conception is an improvement, the meaning and underlying rationale of some aspects of the conception remain unclear (see Figure 4-3). In particular, :

(a) the translation of Jones' 'man-machine interface design' into 'man-machine workstation design' may not be appropriate since the scope of the latter usually comprises the anthropometric design of workspaces. Thus, it is unclear where user interface design is addressed. Further confusion may arise from Shackel's exclusion of software and hardware design from the scope of human factors design (these concerns have been identified as 'machine factors');
(b) the representation of task description, analysis and synthesis as distinct concerns is unexplained. While a case may be made for distinguishing

between task analysis and task synthesis,[19] it is unclear why task description would not be a sub-activity of the two;

(c) the purpose of the 'Relate to' arrow emanating from the 'Other systems' circle is obscure. One interpretation is that it highlights the need to consider extant system characteristics so that transfer of learning effects (both positive and negative) may be addressed. However, the interpretation is

Figure 4-3 : An Updated Conception of the Structured Design of Human-Machine Systems (Shackel, 1986a)



[19] In particular, task analysis can not be conducted in the absence of an extant or reasonably developed system. Thus, to contribute to design analysis for *specification* (as opposed to design analysis for *evaluation*) task analysis needs to be augmented by another step which addresses target system design, namely task synthesis.

inconsistent with other aspects of Shackel's conception. For instance, the suggested location of extant systems analysis relative to task analysis implies that the former is not supported by the latter. The implication would be difficult to support since the strength of task analysis is in extant system analysis. In addition, the location of task analysis seems to imply that the information it generates would not support design decisions on function allocation and task synthesis (see Figure 4-3). Again, the inference would be inconsistent with considerations on transfer of learning effects;

(d) the implications of the 'Regular operation' box for human factors design is unspecified, i.e. its purpose in the design conception is unclear;

(e) the designation of some of the boxes as 'machine factors' is debatable (see Figure 4-3 and (a) above);

(f) the meaning of inter-connecting arrows between the boxes is obscure.

Thus, Shackel's conception needs to be expanded further so that its stage-wise design scope and activities are identified explicitly. Also, the boxes and arrows of the resulting conception should represent at a common level of description, the scope and process of human factors design respectively. The derivation of such a conception is a pre-requisite for the development of a structured method.

To this end, it would be helpful (as an intermediate step) to construct an enhanced conception of structured human factors design as follows :

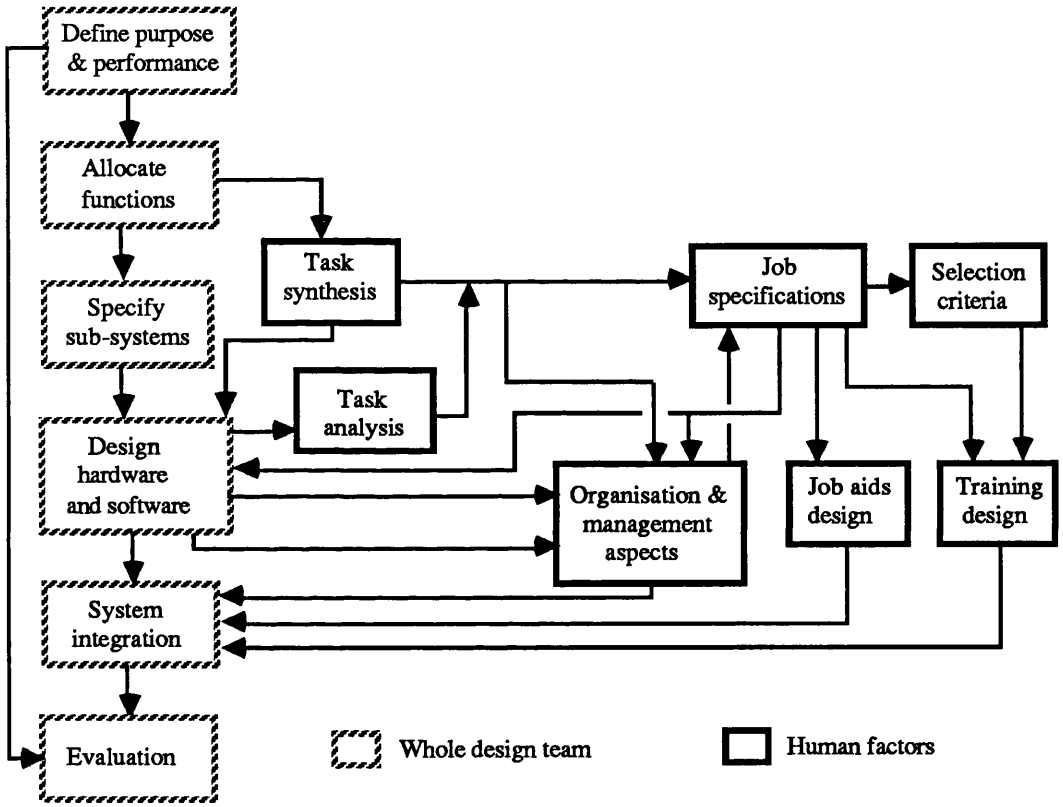
(1) the initial conception of structured human factors design derived previously (Figure 4-1) is carried forward;

(2) an improved version of Jones' and Shackel's conceptions (Figures 4-2 and 4-3) is proposed (Figure 4-4). Since Shackel's conception may be regarded as an extension of Jones' conception, the latter need not discussed further. Modifications implicated by the preceding assessment of Shackel's conception are thus implemented as shown in Figure 4-4;

(3) the conceptions derived in (1) and (2) above are collated into an enhanced conception. Specifically, Shackel's conception (Figure 4-4) is compared with the initial conception (Figure 4-1) and promising aspects of

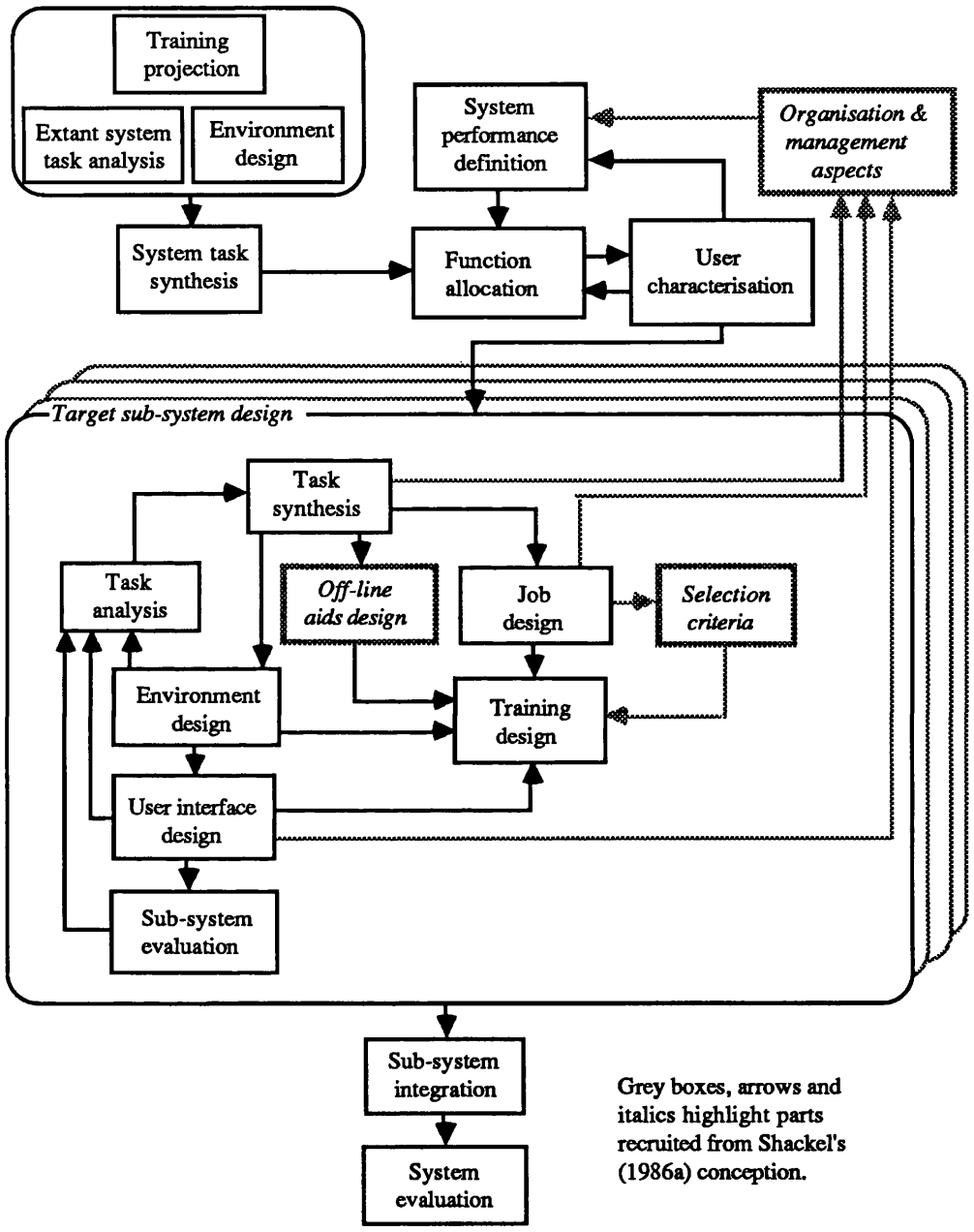
the former are recruited and integrated with the latter, e.g. organisational analysis, personnel selection and job aids (off-line) design.

Figure 4-4 : A Modified Version of Shackel's (1986a) Conception of Structured System Design



The enhanced conception derived is shown in Figure 4-5. Since it describes a reasonably complete human factors design scope, it constitutes the basis on which a structured human factors method may be specified for integration with a chosen SADM (see Chapter 7). Specifically, the scope of the structured method to be developed (and hence the scope of the enhanced conception addressed) is determined by the human factors design support required by the chosen SADM. In other words, the design scope of the SADM is intersected with the enhanced conception to identify a relevant subset of the latter. The subset constitutes a framework for developing a complementary structured human factors method for integration with the SADM. Thus, existing human factors methods and techniques

Figure 4-5 : An Enhanced Conception of Structured Human Factors Design



are examined and recruited appropriately to the construction of a structured method (see Chapters Six and Seven).

To complete the background information associated with this class of research, a review of previous reports on human factors integration with SADMs follows.

Chapter Five : A Review of Previous Research into Human Factors Integration with SADMs

".....And urge the mind to aftersight and foresight."
*T.S. Eliot, 1888-1965, **Little Gidding.***

In this Chapter, previous research into the integration of human factors with SADMs are reviewed. The review covers three SADMs, namely the Jackson System Development (JSD) method; Structured Systems Analysis and Design Method (SSADM); and Yourdon Structured Method. The objectives of the review are as follows :

(a) to note the problems encountered by previous research in the area. Potential pitfalls may then be avoided. Similarly, important research requirements are highlighted for address by the present research;
(b) to incorporate relevant outputs of previous research into the current research (with enhancements as necessary);
(c) to locate the current research against similar work to facilitate later assessment.

An overview of the review follows.[20]

5.1. Human Factors Integration with the Structured Analysis and Structured Design (SASD) Method

This research was carried out by Hakiel and Blyth (1988, 1989) in response to the

[20] The present account does not reflect the order of the SADMs reviewed during the research. Instead, the presentation is sequenced to support the logical development of the thesis. For example, although the work of Carver et al (1987) was reviewed first (since it constitutes the predecessor of the current research) it is presented last (since it sets the context for Chapters Six and Seven).

commercial requirement (Plessey) for a human factors method for user interface design. An additional requirement was that the human factors method should be compatible with their organisation's in-house system design method (namely, the Structured Analysis and Structured Design (SASD) method, otherwise known as the Yourdon Structured Method). The objective of their research was to extend the scope of the SASD method to include human factors design. A human factors method was thus derived. The method will be described following a review of the main concerns addressed by the research.

As in most SADMs, a key activity of the SASD method is the derivation of a context diagram to describe the Essential System Model. The model defines events to which the system must respond and specifies data flows from the system to external terminators. In addition, the model also determines implicitly the scope of subsequent analysis and design activities. Unfortunately, in the SASD method (as in most SADMS) users are frequently located outside the system boundary, e.g. users are described in the context diagram as 'external event generators'. Thus, at a very early stage of system development, the design scope is already confined inappropriately to the specification of a *computer* system rather than a *human-computer* system (Hakiel and Blyth, 1990a). The implication of such a restricted design scope would be an ineffective system since user tasks would not be addressed sufficiently to support the identification of appropriate system functionality and usability. In particular, the exclusion of users from the Essential System Model implies that a particular function allocation (both between users, and between users and machines) is assumed prior to an analysis of overall system performance. Consequently, design considerations on human-computer interaction tend to be expressed only in terms of the event list and data elements of the context diagram. Interaction specification is thus subsumed in the design of communicating sub-systems, i.e. machine-centered design. Such a design perspective would be biased towards the premature definition of low level input and output operations. Since the operations are specified in the absence of a task context, appropriate considerations are precluded on how human-computer interaction may be supported more effectively. As a result, user interface design is restricted to the optimisation of individual displays.

To obviate the observed problems, Blyth and Hakiel (1989) assert that system design should begin with task analysis. On the basis of the analysis, design decisions are then made with respect to the functional deployment of resources (at the organisational and individual level) and the exploitation of new technologies. In other words, early system design activities should comprise the following :

(a) decomposition of system goals into system tasks;
(b) conceptual definition of a system model;
(c) further decomposition of system tasks to support function allocation;
(d) function allocation and description of human tasks, computer tasks and collaborative tasks;
(e) specification of context diagram terminators and an event list.

It was suggested that the above design approach ensures a more accurate Essential System Model which may then be used to constrain system design. In this way, the expected human-machine performance may be ensured.

These suggestions have since been extended into a method via a review of human factors literature (see Figure 5-1). The method structures system design into eight sequential levels of description; namely goal, task, conceptual, semantic, syntactic, lexical, alphabetic and physical levels. Their design scope are as follows :

(a) Goal and Task levels : detailed analysis of system goals and tasks, and function allocation. The design output is a set of task models deriving from various abstractions of the domain (see steps 1 to 4 later);
(b) Conceptual level : identification of conceptual objects that should be represented at the user interface (see step 5 later);
(c) Semantic level onwards : top-down specification of human-computer interaction.

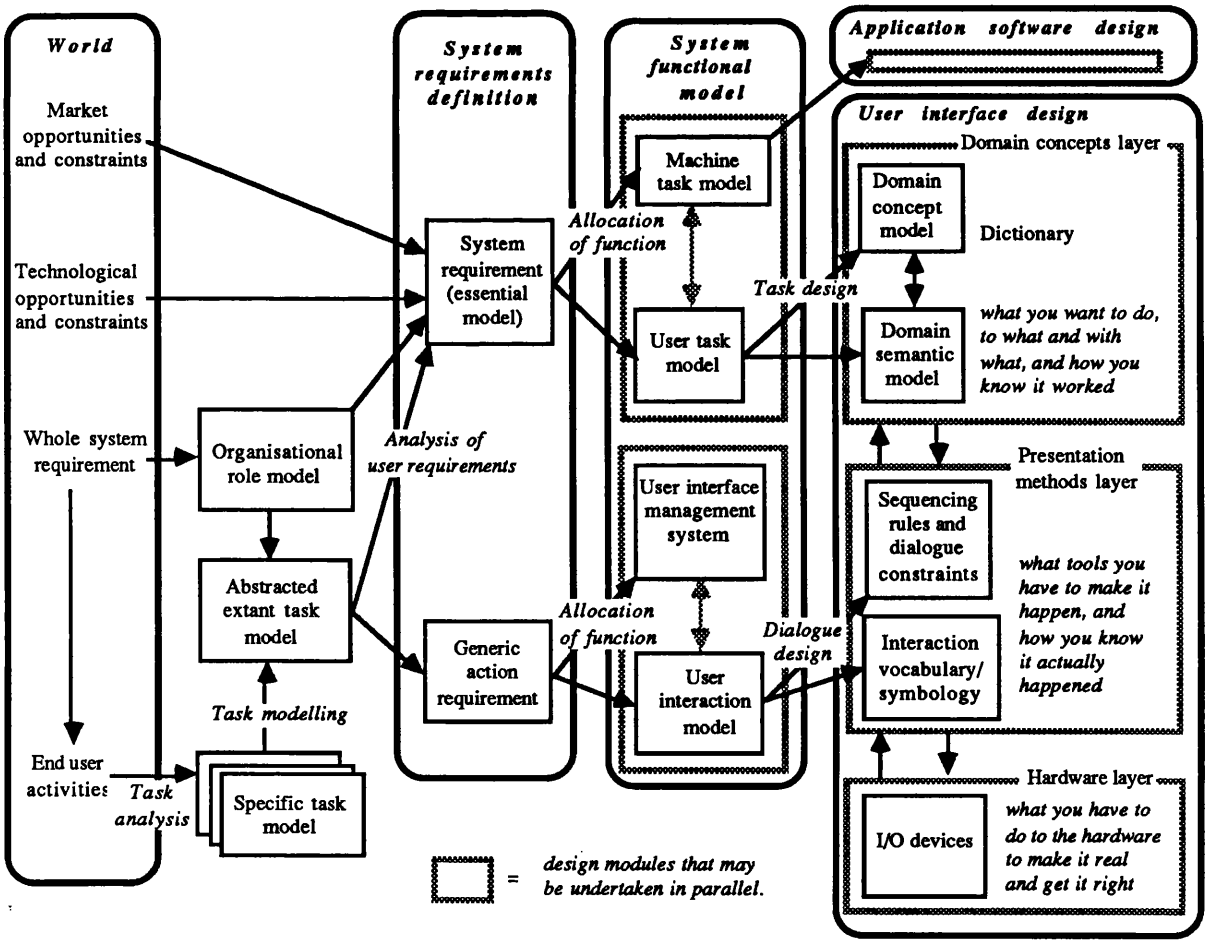
A step-wise description of the method follows.

Step 1 : Analysis of extant systems

(a) identify functional goals of the target system by consulting the initial statement of requirements;

(b) identify existing human-machine systems which share some or all of the target system goals. Real world tasks which map onto target system goals may then be derived and analysed systematically by sampling across a range of existing systems. Technological equivalence between existing and target systems is not required as the subsequent creation of an abstract task model would remove all implementation level details. Where computer-based

Figure 5-1 : Schematic Representation of a Human Factors Method that Complements the SASD Method (Blyth and Hakiel, 1988)



support is already in use, both computer and human sub-tasks should be described.

Blyth and Hakiel's (1989) suggestion to analyse existing systems in general (see (b) above), may be contrasted with the common system development practice of focusing solely on the current system, i.e. the system currently used by the client organisation. The motivation for the suggestion was unfortunately left implicit. Nevertheless, further insight on the suggestion may be found in Lim (1986, 1988d) where a similar analysis of extant systems was proposed[21] as a means of :

(1) ensuring wider consideration of design alternatives. Alternatively, it obviates premature commitment to a design solution (i.e. 'blinkered' design) and the tendency towards the replication of the current solution;
(2) assessing the possibility of wider transfer of learning effects (both positive and negative).

Following extant systems analysis, Blyth and Hakiel's method involves the derivation of task models at various levels of description. To support these descriptions, comprehensive definitions of goals, tasks, roles, jobs, functions, and plans were proposed. Their definitions, as well as those suggested by Dowell and Long (1989), have contributed to a hierarchical taxonomy of work (see Annex C).

Step 2 : User characterisation

(a) identify different classes of users and their particular goals and tasks;
(b) derive the system perspectives corresponding to each class of users.

[21] Some aspects of Blyth and Hakiel's (1989) method are similar to the method described in this thesis. The authors have acknowledged the contributions of the present work, e.g. Blyth and Hakiel (1989).

Step 3 : Task analysis using the Task Analysis for Knowledge Based Description (TAKD) technique (Johnson et al, 1984)

Blyth and Hakiel selected TAKD since it shows the greatest promise as a task analysis technique for supporting the design of human-computer systems. For instance, the following TAKD procedures were adapted for their method :

(a) identify a representative set of tasks from an analysis of existing users and systems. Specifically, the roles and jobs of users should be explored to define a typical set of tasks;
(b) analyse the tasks and describe procedures, goals, sub-goals, etc., textually. The information should be elicited from interviews and observations rather than manuals since the latter tend to describe what users should do rather than what is actually done;
(c) extract a list of all objects (nouns) and actions (verbs) from the task descriptions. These objects and actions constitute the basis on which a conceptual model of the target system may be derived;
(d) apply TAKD techniques (generification in particular) to derive generic actions and objects. Using these generic descriptors, re-express the task description as Knowledge Representation Grammar (KRG) sentences.

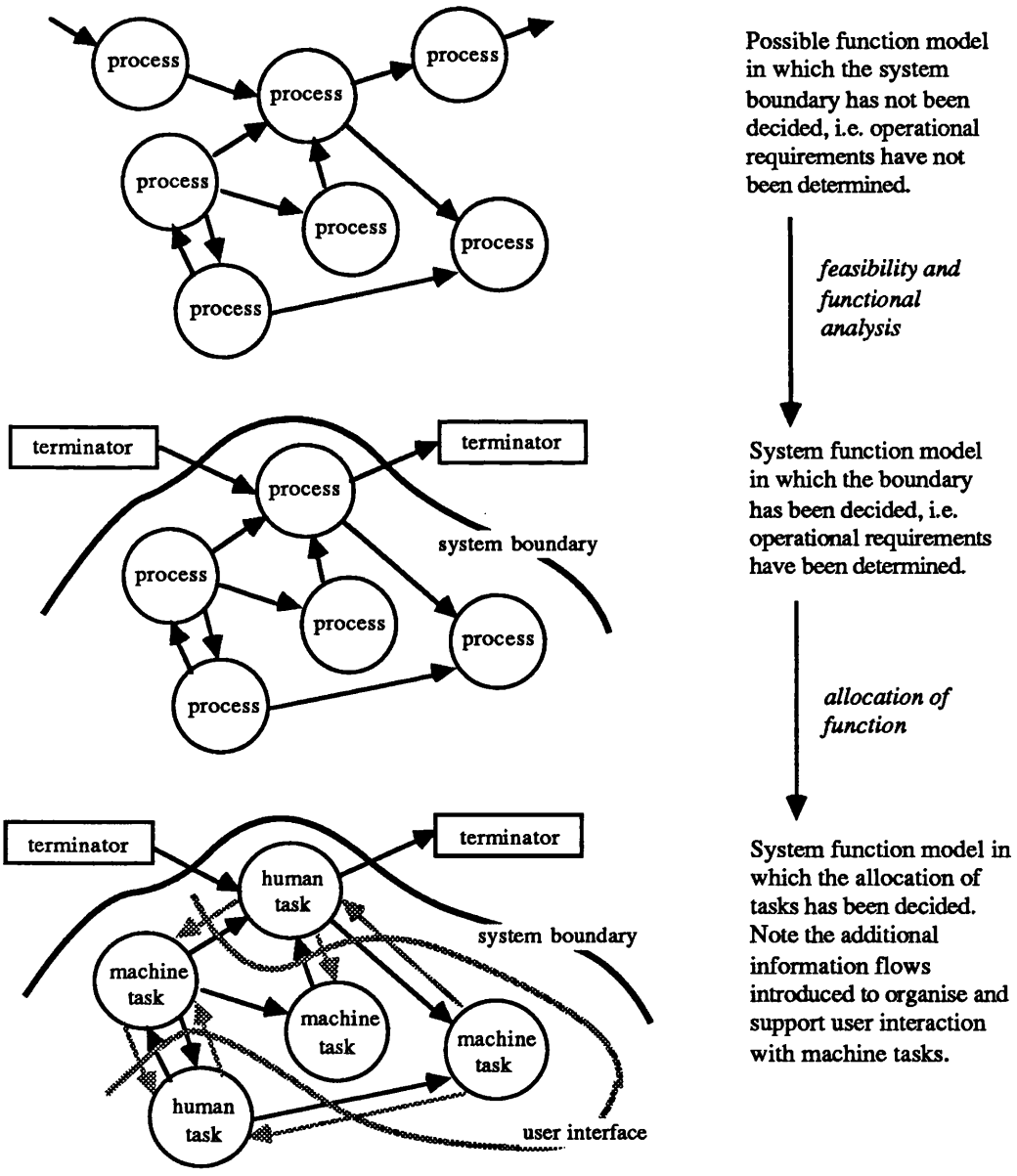
Step 4 : Allocation of Functions

(a) the existing task model derived in Step 3 is incorporated with new requirements and Software Engineering views of target system functionality. An essential system function model with an explicit boundary is thus defined;
(b) function allocation between the human and computer may then be considered iteratively to meet the performance requirements of the overall system (see Figure 5-2). Since existing performance data may not provide adequate support, prototyping might be necessary in determining an appropriate allocation of function.

Several weaknesses in this step of the method are noticeable, namely :

(1) the relationship between KRG descriptions of the existing task model and the processes in Figure 5-2 is unclear, e.g. would one KRG sentence be equivalent to a process ?

Figure 5-2 : Function Allocation using a System Function Model[22]



[22] Figure reproduced from Hakiel and Blyth (1990a).

(2) previous weaknesses of TAKD remain unaddressed. For instance, it is unclear how KRGs are organised to define task structures and user roles. Yet, the task model is required to make three crucial contributions to function allocation, namely : to ensure job, role and task coherence; to provide an explicit account of potential workloads; and to ensure that information flows across the user interface are appropriately contextualised to task requirements. In the absence of further developments, these contributions would not materialise.

In contrast to the derivation of an essential system function model, function allocation concerns were considered in much greater detail. For instance, two function allocation techniques, namely those proposed by Price (1985) and Clegg et al (1989), were identified by Blyth and Hakiel (1989) and Hakiel and Blyth (1990b) respectively. Since these techniques appear to be promising, they were noted for further investigation (see Chapter Seven).

At this step of the method, the performance statements should be specified in an appropriate format to facilitate the assessment of alternative allocations of function. Specifically, the format should comprise the following :

Who ? e.g. user characteristics
Doing what ? e.g. nature of task
What circumstances ? e.g. work environment
What performance ? e.g. worse and best case for a particular prototype
How performance is tested ? e.g. full system scenario.

Since the format satisfies ISO's proposal that requirement specifications should be linked to testing schemes (ISO/TC 159/SC 4/WG 5 N84), it was noted for later comparison with similar proposals by other researchers, e.g. Whiteside et al (1985).

Step 5 : Defining the conceptual model of the system

The conceptual model of the system comprises a set of objects and user actions that

the system must support. It is derived as follows :

(a) task objects are separated into classes and sub-classes and their properties are defined. For each object, actions that affect its properties are defined. A scheme for documenting these descriptions is included in the method (see Blyth and Hakiel, 1988, 1989). These design activities correspond to those found in the modelling phase of the Jackson System Development (JSD) method;
(b) compositional and taxonomic relations between objects that comprise the domain knowledge of the user, are described using a network diagram. Such diagrams are used for similar descriptions in the development of knowledge based systems (see Ragoczei and Hirst, 1990). Since the description represents a useful summary of domain semantics, it was noted for further consideration during the specification of a human factors method to complement the JSD method (see Chapter 7);
(c) a specific task model is generated to describe how objects and actions of sub-tasks may be composed into higher level tasks. It appears that the concept of a specific task model was recruited from the Knowledge Analysis of Task (KAT) technique (Johnson and Johnson, 1988). However, its present purpose is unclear since higher level tasks would have been specified in Steps 3 and 4.

Subsequent to this step, user interface design at the semantic, syntactic, lexical, alphabetic and physical levels is undertaken. However, little information is available on these design activities. Hence, no further comment is possible on the rest of the method.

In conclusion, Blyth and Hakiel's work failed to include an explicit integration of the human factors method with the SASD method. In particular :

(1) design inter-dependencies and the timing of design processes between the two methods were not made explicit;
(2) the possibility of using a common notation was not considered.

Also, some steps of the method are poorly described or incomplete. Thus, the potential contributions of their work with respect to the present research, are restricted largely to :

(1) the identification of promising human factors techniques, e.g. function allocation and TAKD;
(2) the specification of human factors definitions, e.g. tasks, functions, etc.;
(3) the identification of promising design description schemes, e.g. formats for performance specification and domain description.

5.2. Human Factors Integration with the Structured Systems Analysis and Design Method (SSADM)

This consultancy based project (200 person-days) was undertaken by HUSAT (Loughborough) for the Department of Health and Social Security (DHSS). Its aim was to extend DHSS's in-house version of SSADM called DIADEM (Departmental Integrated Application Development Methodology). The result is a conglomeration of SSADM, PROMPT (Project Management Technique) and a human factors method that may be used by designers with little human factors training. The latter method constitutes the focus of this sub-section.

HUSAT's human factors method adopts a participative approach and its scope was intended to cover both human and organisational design.[23] However, the original scope was curtailed as only four out of the following areas of human factors design concerns (namely those in bold italics below) were targeted by the client for incorporation into DIADEM :

(1) ***user analysis*** and socio-technical systems analysis

[23] The scope of the project also includes collating reference manuals to support users of *extended* DIADEM who are not trained specifically in human factors. Thus, the manuals describe declarative knowledge associated with the selected areas of human factors design concerns (see above). Since the manuals are not available, no further discussion is possible.

(2) user involvement in decision making

(3) user acceptability criteria

(4) methods of user involvement

(5) ***job design*** and work organisation

(6) ***task allocation/job stream charts***

(7) human computer interface design

(8) ***prototyping***

(9) workplace and workstation design

(10) user support

(11) management of change

(12) institutionalising human factors

The objectives of the project were to specify and locate design activities and procedures of the above design concerns around DIADEM (see Figure 5-3). Unfortunately, detailed information on most of these design concerns is not available as the work is crown copyright. The relatively few published reports on the project[24] provide only partial coverage of the work. As such, it is unclear how design is operationalised between levels which are far apart, e.g. moving from : task allocation and job design ---> work organisation ---> dialogue design (see Figure 5-3, right-hand section). Although a set of lower level design products have been specified (Figure 5-3, middle section), the information is still insufficient to support an assessment of the method, e.g. its overall coherence and completeness. Thus, only two of the above design concerns, namely user analysis and task allocation charts (i.e. (a) and (f) respectively), will be reviewed presently. These design concerns were reported in Damodaran et al (1988) and Ip et al (1990) respectively.

In the context of the method, user analysis comprises the following design

[24] The present review was collated from a seminar presentation (Damodaran, 1988) and two conference papers (namely Damodaran et al,1988; Ip et al, 1990).

activities :

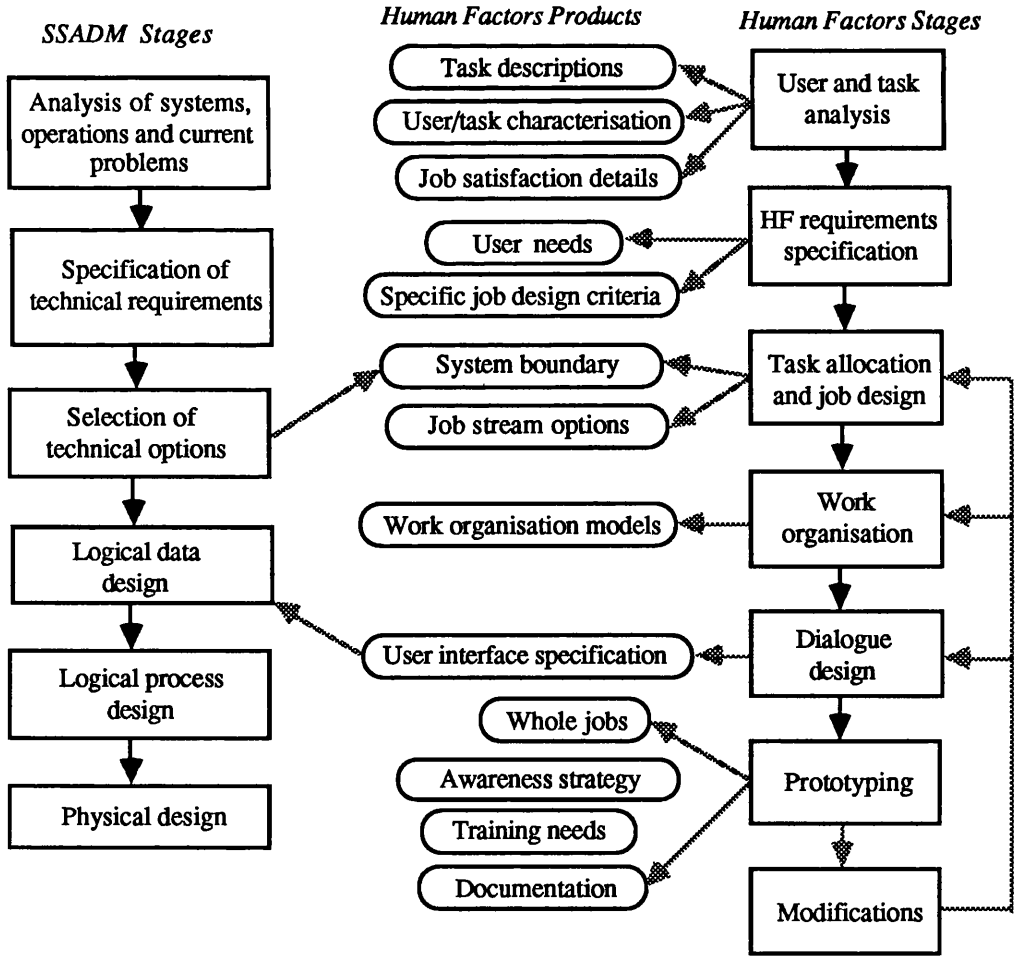
(1) user classification

(2) job and task characterisation

(3) work role analysis

The set of design checklists, questionnaires, observations sheets and description summary forms for user analysis is shown diagrammatically in Figure 5-4. Unfortunately, no further comment can be made since detailed information is unavailable.

Figure 5-3 : Schematic Representation of a Human Factors Method that Complements DIADEM (Damodaran et al, 1988)



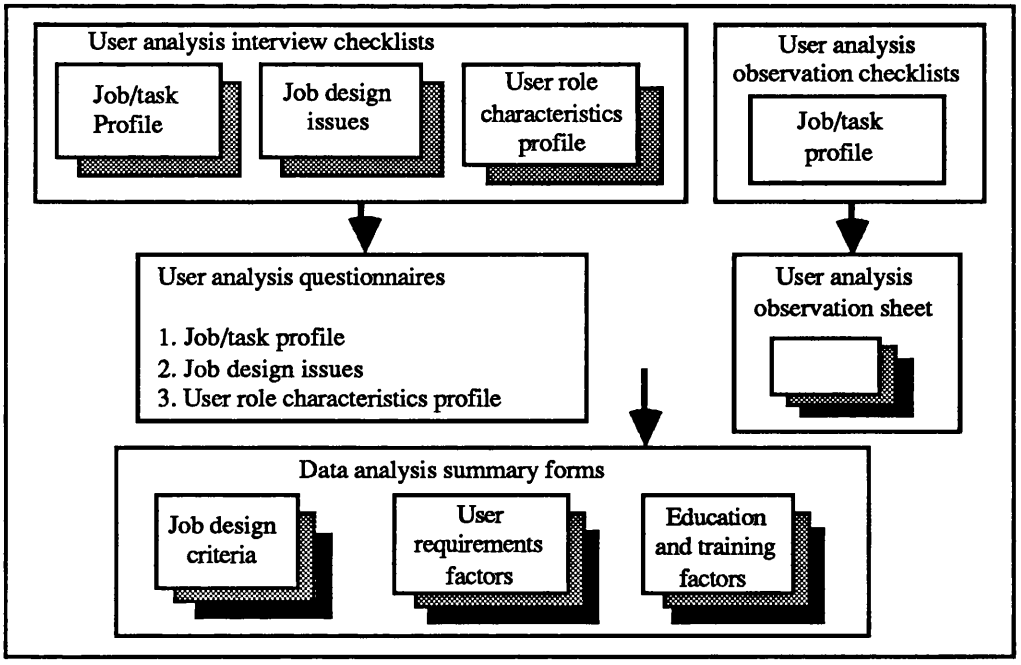
The second human factors concern of the method addresses task allocation. In particular, a task charting technique was proposed to support requirements specification. Specifically, the technique supports the following :

(a) definition of the computer system boundary. The definition involves an exploration of alternative human-machine task allocations and their impact on users' jobs;

(b) definition of functional requirements of alternative designs derived in (a) above. The system boundary is then finalised by selecting a particular configuration of human-machine task allocation, e.g. functional requirements for the on-line task are detailed. These design decisions precede the Logical Design Stage of DIADEM;

(c) communication of alternative designs to different end-user groups. For instance, manual job design alternatives may be explored. Since various job roles are discussed with users, their feedback may be more completely incorporated into the design. Thus, the selection, integration and evaluation

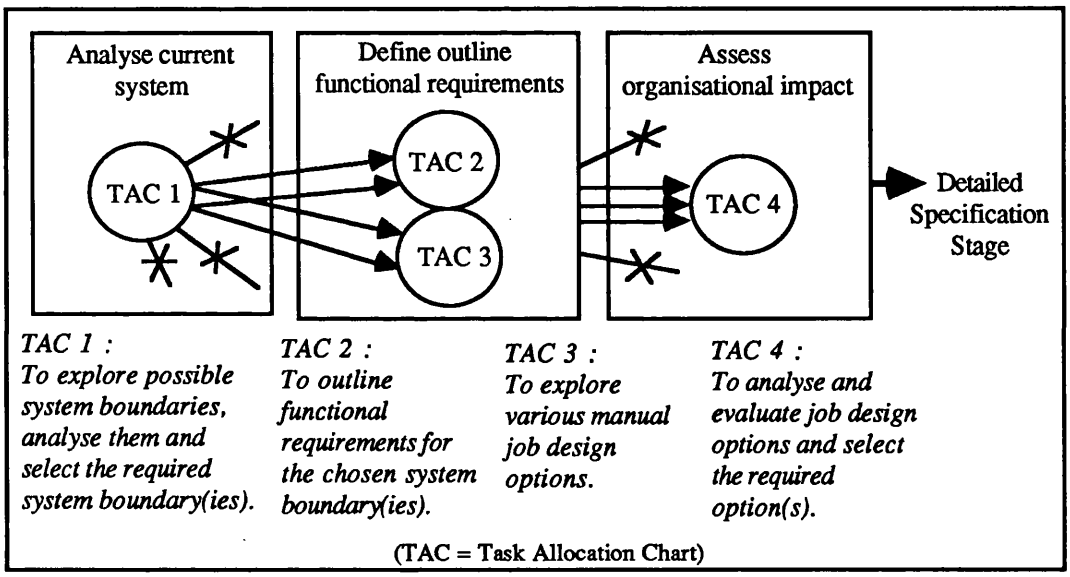
Figure 5-4 : Checklists, Questionnaires, Observation Sheets and Summary Forms for User Analysis in DIADEM



of on-line and manual job design alternatives are facilitated. Appropriate design alternatives can then be elaborated into user requirement specifications to constrain the design of the automated system.

The steps of the task charting technique are summarised diagrammatically in Figure 5-5 below.

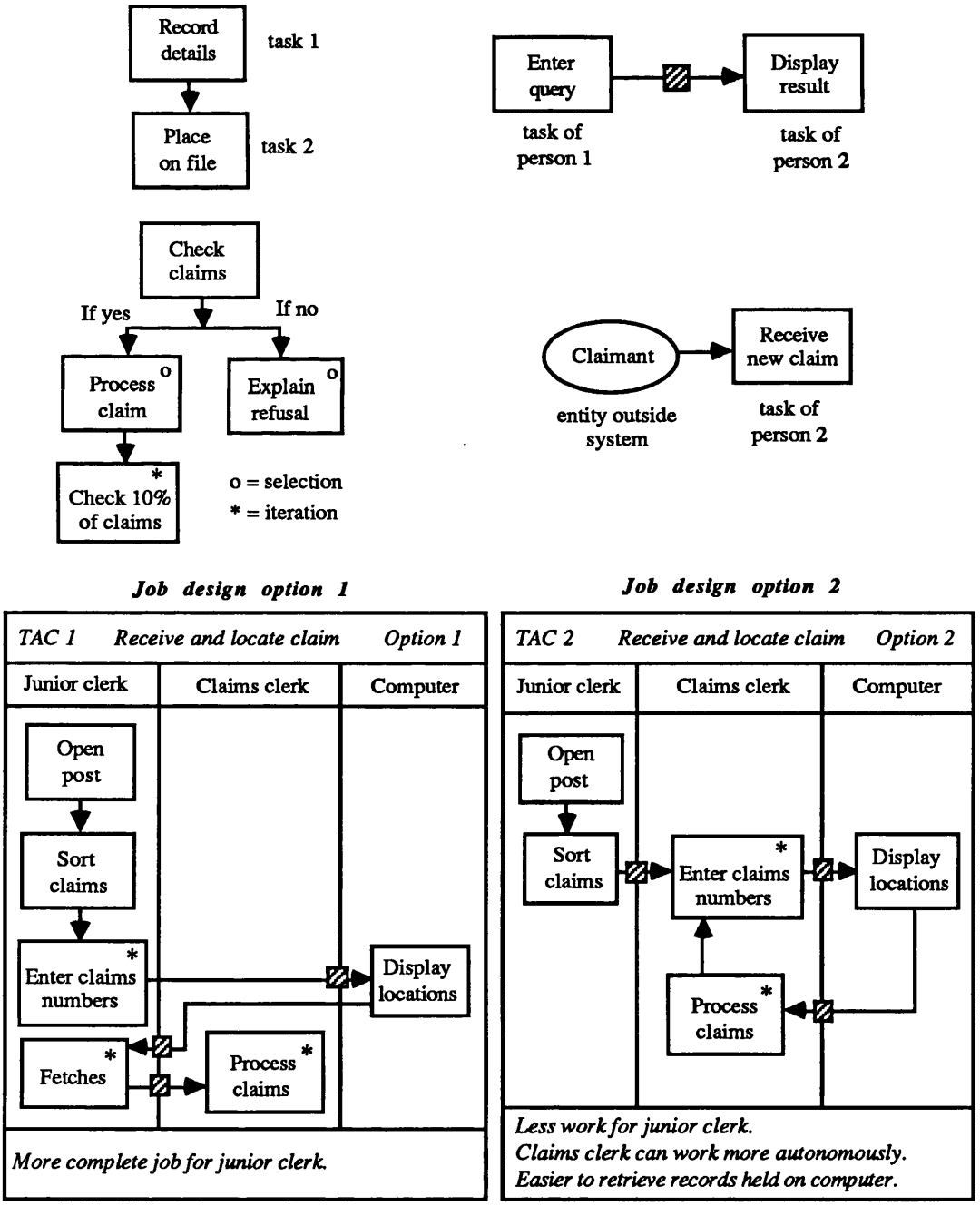
Figure 5-5 : Design Steps of the Task Charting Technique



The notation for describing alternative job designs is shown in Figure 5-6. This notation was preferred over data flow and entity life history diagrams since it was considered more appropriate for end-users. In particular, task allocation charts provide visual representations of the procedures of task functions. In addition, they can be used to link proposed screen formats to textual descriptions of the functions. Thus, it was asserted that the task charting technique is particularly useful for highlighting relationships between users' jobs, and dialogue and screen design. However, several problems with the technique have been reported by Ip et al (1990).

Firstly, the documentation of task flow charts is cumbersome. Specifically, chart drawing is a time-consuming and labour intensive task. In addition, the descriptions that are produced require an inordinately large volume of paper. These problems are compounded further by the need to produce charts at various levels of task flow description, e.g. job and work organisation levels.

Figure 5-6 : DIADEM's Task Allocation Chart Notation



Secondly, at the time of the project, it was found that users of the technique could not identify the relationship between the charts and design outputs of DIADEM. Ip et al (1990) reported in retrospect that the charts may be linked to events associated with one or more entities of DIADEM. In other words, particular sequences of user task events should intersect with the Events Catalogue (a DIADEM output). Similar observations have been made by Carver et al (see later) with respect to users' tasks and entity actions modelled in the Jackson System Development method.

In conclusion, the DHSS project failed to specify explicit design inter-dependencies between the human factors method and DIADEM. Specifically, a high level conception of how human factors activities should be located against DIADEM stages, would not be specific enough to support the definition of design inter-dependencies. Furthermore, there is little evidence to suggest that a structured human factors method was derived for integration with DIADEM. Instead, the project seems to have concluded with the assignment of a set of discrete human factors techniques against the methodological framework of DIADEM. In the absence of detailed publications and development of the method, the contribution of the DIADEM project to the present research has been limited.

5.3. Human Factors Integration with the Jackson System Development (JSD) Method

Two attempts have been made at integrating human factors with the Jackson System Development (JSD) method, namely those undertaken by Carver et al (1987) and Sutcliffe (1988a, b). Presently, their contributions are reviewed in turn.

5.3.1. The Work of Sutcliffe (1988a and b)

Sutcliffe's work was motivated by his belief that the :

> *".....practice of good human-computer interface design will only result*

from the integration of Human-Computer Interaction principles and procedures within existing system design methods, rather than by the creation of stand-alone Human-Computer Interaction methods."

Although his motivation is consistent with the present research, the scope of his work primarily comprises the location of disparate human factors techniques at particular stages of the JSD method. In other words, no attempt was made at developing a *structured human factors method* to complement the JSD method. Thus, the present review is confined to the human factors considerations and techniques that were recruited to support an extended JSD method. Specifically, the scope of the method was extended to include *task analysis* and *user-computer dialogue specification* (Sutcliffe, 1988a).

In respect of *task analysis* and the JSD method, three human factors considerations were highlighted. Firstly, the complexity of initial task descriptions should be analysed to support decisions on the allocation of on-line, off-line and automated tasks. Specifically, Sutcliffe suggested that complexity analysis would support :

(a) design analysis and modifications. For instance, an unacceptably complex task would indicate that smaller sub-task elements need to be specified. Thus, appropriate computer functions and displays may then be designed to support the on-line tasks;
(b) an appropriate matching of user skills and task complexity;
(c) design decisions on an appropriate variation of task complexity within an individual user's work.

It was also suggested by Sutcliffe (1988a) that a simplified application of the Cognitive Complexity Theory (CCT) would enable such an analysis. The simplification is motivated by the concern that CCT as it stands is much too complicated for direct application by designers. These suggestions were then investigated in a student trial. The results indicated that the expected benefits of complexity analysis did not accrue (Sutcliffe, 1988a). The negative outcome may be

explained by the following :

(a) the subjects found the complexity metrics difficult to interpret and use in spite of the simplications proposed by Sutcliffe (see (b) below for an example). Thus, the simplified technique was not well received. Further comment on the technique is not possible in the absence of a more complete description in the reports;
(b) the results derived from applying the technique were imprecise. The latter may be attributed to the assumption that subjectively assigned complexity units may be summated to determine the overall complexity of the task. The validity of such assumptions is presently undemonstrated;
(c) the task of translating assessment results into an appropriate design expression is almost completely dependent on craft knowledge and designer expertise.

Thus, the proposed analysis technique can not be considered as promising.

Secondly, in contrast to a recommendation of the JSD method, Sutcliffe emphasised that links between the JSD entity model and the user's task should be made explicit. Specifically, task activities constitute processes which interact with JSD entities. Thus, the scope of JSD interactive functions could be extended to link task processes with the JSD model as the central focus.[25] For instance, the user's task may be related to relevant attributes of the entity (e.g. for the book entity, an example may be state = 'reserved' and location = 'shelving stack'). These suggestions by Sutcliffe are similar to those of Carver et al (1987) as described in the next sub-section.

Thirdly, Sutcliffe suggested that JSD structured diagram notation could be used to describe tasks so that design communication problems may be avoided by the use of a common language between the disciplines. Again, a similar suggestion may be found in Carver et al (1987). However, Sutcliffe proposed a novel interpretation of

[25] This may be contrasted with user-centered design for which the focal point is the user's tasks.

the selection construct when found in a task description. He asserted that these incidences would correspond to logical break-points in task execution, and that they should be considered as task closures during display design. Unfortunately, an illustration on the influences of task closures on user interface design was not provided. Thus, apart from the specification of computer response times their intended application remains unclear.

Sutcliffe's extension of the JSD method also includes four suggestions on *user-computer dialogue* specification, namely :

> (a) entity actions should map onto permissible actions on objects at the user interface since the JSD entity model is essentially an object and event model. For instance, in a library application, actions carried forward from the JSD model to the user interface may include 'Acquire' and 'Archive' a book. However, Sutcliffe's suggestion is less specific than the proposals of Carver et al (1987). Specifically, the latter highlighted that JSD entity actions do not necessarily correspond on a one-to-one basis with the actions of user interface objects. This observation was not revealed by Sutcliffe's case-study as it was too simple;
> (b) previous functional support specifications for the user's task should comprise the basis for user-computer dialogue specification. However, the suggestion was not expanded. Although a case-study was considered, the structured diagram descriptions of the user's task failed to illustrate how user interface design may be constrained; [26]
> (c) the set of JSD filter processes constitutes a framework for user interface design (Sutcliffe, 1988b). It was suggested that the user-computer dialogue may be derived by extending JSD input sub-system specifications since they describe user inputs and errors with respect to the JSD model (i.e. its

[26] The inadequacy is rectified by a later paper, namely Sutcliffe and Wang (1991). The paper was excluded from the present review of previous research contributions to the thesis because many ideas from this thesis were recruited by them (rather than vice versa). However, a comparison between the thesis output and their overall research output is described in Chapter Twelve.

actions and entity attributes -- also see (a) above). For instance, dialogue control requirements for error recovery, prompts and feedback messages, may be detailed as *separate* simple filter process descriptions. Although the descriptions are consistent with the JSD method, they provide only a *segmented* view of the interactive task (i.e. as opposed to a coherent whole). Thus, Sutcliffe suggested that inputs of filter processes should be grouped into transactions in accordance with the user's view. However, no example was provided on how the grouping should be described. Fortunately, a similar suggestion was proposed in Walsh et al (1989) where it was suggested that filter process inputs should be sequenced using one JSD structured diagram (more structured diagrams if concurrent processes are involved). Thus, relevant feedback from users may be elicited to derive a user-centered view of the interactive task;

(d) JSD structured diagram notation should be used to describe user interface objects, e.g. to relate permissible object actions and roles.

In conclusion, Sutcliffe's suggestions are focused on extending the JSD method. His objectives did not include the specification and integration of a structured human factors method with the JSD method. In particular, his proposals did not include an adequate account of the scope and process of human factors design. As such, the design relationships identified between human factors and JSD is confined predominantly to the scope of the latter, e.g. few locations and intersections of JSD and human factors design stages were identified. Consequently, Sutcliffe's suggestions constitute a less specific subset of the proposals of Carver et al (1987).

5.3.2. The Work of Carver, Clenshaw, Myles and Barber (1987)

Carver et al (1987) reported an informal attempt at integrating human factors with the JSD method. The attempt was motivated by the following observations :

(1) inadequately specified systems were extremely difficult to modify;

(2) significant benefits would accrue from earlier and closer collaboration between human factors designers and software engineers. Thus, the problems observed in (a) above may be obviated.

Their observations derive essentially from experiences of design projects commissioned by the Ministry of Defence, e.g. experiences on the Divisional Intelligence (G2) All Sources Cell project during military exercises 'Crested Eagle' and 'Lionheart'. The following objectives to provide human factors support for the JSD method were thus defined :

(a) to improve the usability of the JSD method by enhancing existing procedures;
(b) to extend the design coverage of the JSD method to include the specification of the human-computer interface;
(c) to explore the possibility of documenting the design process in machine readable form.

In respect of objective (a) above, two proposals were made, namely :

(i) the procedure of extracting nouns and verbs from interview transcripts to identify candidate entities and actions respectively should be modified. Carver et al (1987) noted that the current procedure of constructing and refining *independent* lists of nouns and verbs resulted in the loss of useful information since verbs were extracted out of context. Thus, they proposed that candidate actions should be collated by listing events in terms of their verbs, subjects and objects. In this way, a list of entities and actions could be extracted more effectively;
(ii) requirements definition should be represented explicitly prior to the modelling stage of the JSD method. Specifically, the information elicited from interviews and existing documentation should adequately support : the formulation of user needs statements; the identification of any hardware constraints; the definition of a requirements statement on the types of facilities that should be supported by the system.

These proposals were targeted at *enhancing* procedures of the existing JSD method. In contrast, objective (b) above is targeted at *extending* the design scope of the method. For instance, its scope could be extended to include project selection, cost and benefit analysis, project planning and management, and user interface specification. Due to resource constraints, Carver et al (1987) focused on an extension of JSD to include user interface specification. In this respect, they suggested the following :

> (i) JSD specifications during the initial model stage should explicitly address human-computer interaction. In particular, it was noted that preliminary specifications of the input sub-system should be linked to user interface design since the former interposes between the real world and the JSD model. Thus, screen display entities that intervene between the JSD model and the real world (via the user) may be defined (see Figures 5-7 and 5-8). Interactive facilities supporting the processes and connections associated with the display entities are then detailed in the information function stage. In this way, a list of on-line interactions with the facilities may be linked to

Figure 5-7 : JSD Initial Model Extended to Include a Human-Computer Interaction Layer (Carver et al, 1987)

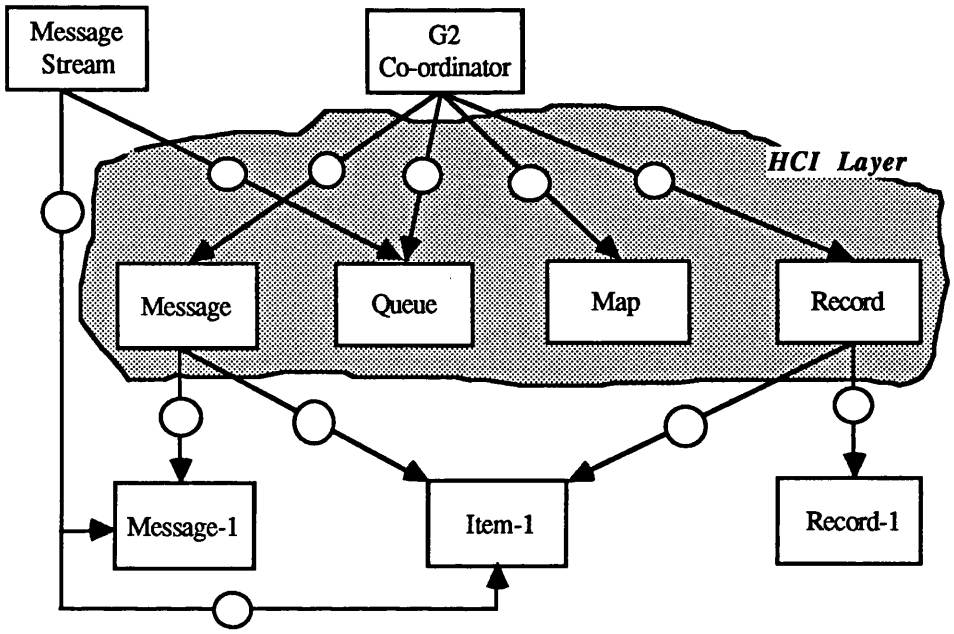
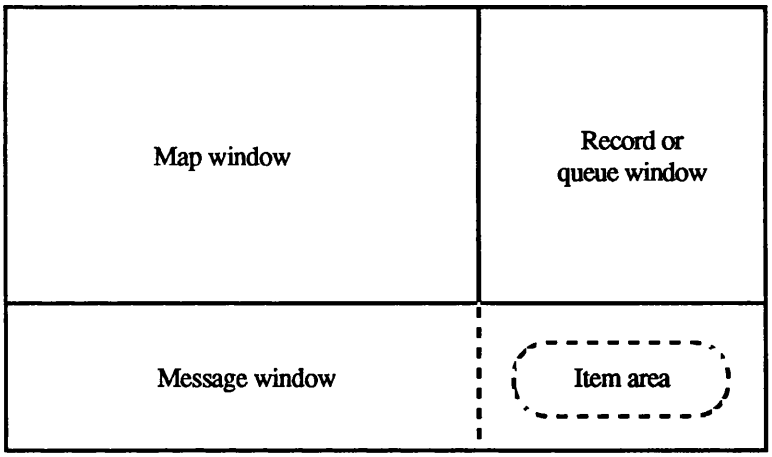


Figure 5-8 : Screen Layout Description Corresponding to the Extended JSD Initial Model in Figure 5-7



specific actions of the JSD model. It should be noted that such linkages would not always comprise a one-to-one relationship. Display design is then undertaken based on the described interactions;

(ii) task and goal analysis should be recruited to the method to clarify the purpose of the computer system and the functional support it should provide in respect of the user's task.

To address the above concerns, Carver and Cameron (1987) suggested the following high level procedures for user interface specification in the extended JSD method :

(i) build the JSD model;

(ii) identify functions in the network phase. A basic set of inputs (one for each action and enquiry) is thus defined;

(iii) associate the inputs with users' tasks;

(iv) define the interactions required for each input, and express the interactions as a combination of actions on the user interface;

(v) introduce further functions to support the required interaction and the users' task;

(vi) specify off-line events to complete the description of the users' task.

The final objective of Carver et al (1987) was to document the user interface specifications in a machine readable form. In this respect, it was observed initially that the JSD structured diagram notation could not describe events that are ordered unpredictably; e.g. multiple events that occur haphazardly; and multi-tasking with no pattern of controlling each of the ongoing processes. However, such problems were resolved by extending the constructs of the JSD notation, e.g. to include the description of concurrent events. Thus, it was demonstrated that the notation constitutes a powerful language for task description and user-computer dialogue specification. Furthermore, existing computer tools such as Program Development Facility (PDF), MacDraw™ and FileVision™ may be recruited to support design documentation. Consequently, Carver et al (1987) reported that machine readable documentation was achieved.

To summarise, the work of Carver et al (1987) highlighted the following :

(a) the JSD method is defined sufficiently to support an intersection of its design concerns with those of human factors design (Carver et al, 1987);
(b) the JSD method encourages user centered design since it emphasises real world modelling. Thus, its design approach is compatible with human factors design (Carver, 1988);
(c) the JSD structured diagram notation constitutes a common language for describing Human Factors and Software Engineering design. Thus, inter-disciplinary communication is facilitated (Carver and Cameron, 1987).

In conclusion, the work of Carver et al (1987) may be considered the precursor of the present research (see Chapter Seven, and Chapters Nine to Eleven).[27] To extend their work, the objectives of the present research would include the following :

(i) to derive a more complete and explicit scope and process of human factors design;

[27] To be more specific, their observations culminated in the initiation and sponsorship of the present research.

(ii) to identify a more complete intersection between human factors design and the JSD method on the basis of (i) above;

(iii) to construct a structured human factors method that complements the JSD method;

(iv) to integrate the human factors method with the JSD method.

Since the scope of human factors design is extensive (see Chapter 4, sub-section 4.3), only a subset of the concerns derived in (i) above can be addressed. Thus, the final scope of the present research is re-defined in the next Chapter.

PART III :

On Human Factors Integration with the Jackson System Development Method

CONTENTS

Chapter Six : General Research Concerns for Integrating Human Factors with the Jackson System Development Method.....132

6.1. Choice of the Jackson System Development Method and other Constraints on the Present Research.....132
6.2. An Overview of the Jackson System Development Method.....137
6.3. Identifying the Human Factors Design Support Required by the Jackson System Development Method.....140
6.4. Research Requirements for Human Factors Integration with the Jackson System Development Method.....142
6.5. An Instantiation of the General Research Plan for Integrating Human Factors with the Jackson System Development Method....145

Chapter Seven : Research Milestones in Integrating Human Factors with the Jackson System Development Method.....155

7.1. An Overview of the Research into Human Factors Integration with the Jackson System Development method.....155

7.1.1. A Review of Activities for Specifying Stage-Wise Design Notations and Documentation Schemes of the JSD*(HF) Method.....157
7.1.2. A Review of Activities for Specifying Stage-Wise Design Scope and Processes of the JSD*(HF) Method.....164
7.1.3. A Review of Activities for Specifying Inter-Dependencies between Design Streams of the JSD* Method.....180

Chapter Six : General Research Concerns for Integrating Human Factors with the Jackson System Development Method

".....Thought is the child of action....."

Disraeli, 1826, Vivian Grey.

This Chapter establishes the context of the research activities described in Chapter Seven. To this end, its objectives are as follows :

(a) to set out the constraints of the present research, and expose the rationale for integrating human factors with the Jackson System Development (JSD) method;
(b) to identify the human factors design support required by the JSD method;
(c) to define the research scope and requirements of (a) and (b) above. For instance, the constraints of the present research essentially limits its scope to a sub-set of the human factors design concerns identified in Chapters Four and Five;
(d) to instantiate the general research plan for integrating human factors with SADMs (Chapter Three) with respect to the present research context, i.e. the instantiated plan should address the requirements set out in (c) above.

An expanded account of these objectives follows.

6.1. Choice of the Jackson System Development Method and other Constraints on the Present Research

Two criteria dictated the choice of the Jackson System Development (JSD) method for the present research; namely the appropriateness of the method and the preference of the research sponsor. Since the latter was influenced by the former,

subsequent discussions on the choice of the JSD method are focused on its beneficial characteristics.

In addition to the benefits of SADMs in general (see Chapter Two; Lim et al, in press), the JSD method is also beneficial for the following reasons :

(a) JSD is a well established SADM and is among the more popular methods for developing real-time systems (Morrison, 1988; Wilson et al, 1989);

(b) JSD specifications are in principle directly executable (Carver and Cameron, 1987). Thus, a system produced using JSD may be viewed as a simulation of the relevant parts of the real world and its functions provide outputs of the simulation (Carver, Clenshaw et al, 1987). An emphasis on simulating the real world may provide a means for accommodating an appropriate user's model of the system (see (c) below);

(c) JSD starts by modelling the proposed computer system in terms of objects and events in the real world of the user (this may be contrasted with SSADM which begins by data flow description). Since JSD specifications are developed from an understanding of the user's world, they intersect with user requirements capture, task description and analysis (see Chapter Five, sub-section 5.3). These intersections facilitate the integration of human factors with the JSD method;

(d) JSD has been shown to complement object-oriented methods and is itself object-oriented (Birchenough and Cameron, 1989; Cameron, 1987). Thus, the methodological framework of JSD could potentially contribute to the design of object-oriented user interfaces. Since such user interfaces are reportedly superior in usability characteristics (Selby and Long, 1991) and have made considerable inroads in the personal computer market (e.g. Apple Macintosh), it would be pertinent to consider the integration of human factors with an object-oriented method such as JSD;

(e) the well developed methodological framework of JSD facilitates the location of human factors inputs. For instance, its input sub-system has been used as a framework for locating user interface design (Sutcliffe, 1988b; Carver et al, 1987). In addition, general design concepts of JSD

may be recruited. For instance :

(1) its input sub-system specification provides a taxonomy of input errors, e.g. context error, simple input error, etc.;
(2) its information functions address design semantics, e.g. system outputs are contextualised to the requirements of the JSD model;
(3) its categories of information flow (e.g. state vector inspection and data-streams) and its assignment of time grain markers identify important aspects of information specification, i.e. content; direction of flow; timing, currency and duration of display; etc.

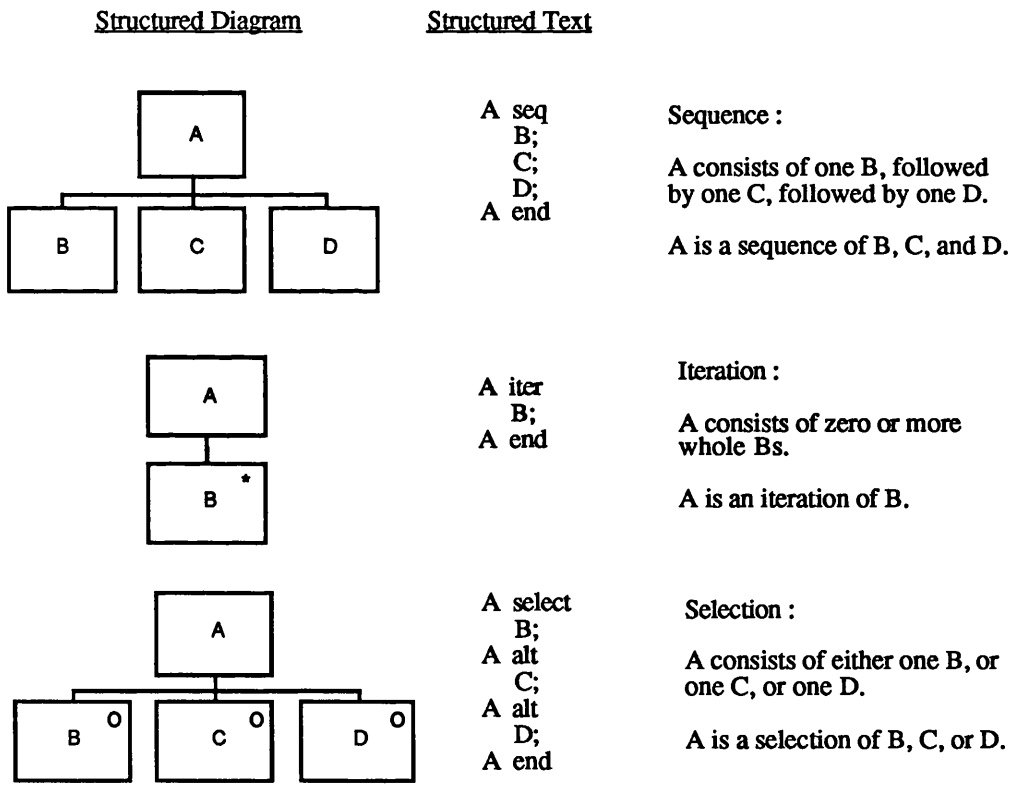
(f) JSD structured diagram notation has been shown to be suitable for precise description of users' tasks (Carver, 1988; Carver and Cameron, 1987; Sutcliffe, 1988a). Specifically, the notation offers well developed constructs which describe sequential, selection, iteration, concurrent, interlocking, backtracking and uncertain events (see Figure 6-1; Carver and Cameron, 1987). JSD notation has also been shown to be a graphical equivalent of BNF (Boldyreff, 1986). This equivalence with an established notation for describing human-computer interaction, not only indicates the capability of the JSD notation for similar description, but additional benefits owing to its graphical nature may also accrue. For instance, the notation has been reported to be easily understood by users (Carver, 1988; Finkelstein and Potts, 1985). Thus, user feedback elicitation is facilitated. Other benefits which accrue from recruiting the notation for human factors descriptions are as follows :

(1) the poor specificity of current human factors descriptions would improve since the notation is more precise;
(2) the use of a common notation would obviate communication problems between software engineers and human factors designers. Since the primary objective of integrating human factors with SADMs is to ensure early and continued human factors involvement (i.e. via collaborative design), effective and unambiguous communication between designers of the two disciplines is essential. A common

language is therefore crucial. Since existing JSD notations also support human factors descriptions (see above), notational integration is facilitated;

(3) *formal* human factors specification may accrue in the future since JSD notations could potentially map onto the notations of formal methods. Such a translation has been demonstrated by Sridhar and Hoare (1985) for CSP (Communicating Sequential Processes).

Figure 6-1 : Basic Constructs of JSD Structured Diagram Notation



These desirable characteristics of JSD exerted a strong influence on its choice for the current research. The choice is also appropriate since no research has been commissioned *specifically* to investigate the integration of human factors with the method. Previous insights on the potential of the integration were gleaned from opportunistic and incidental observations during system design and consultancy projects. In other words, previous efforts were not directed at method development.

A contrary situation applies for SSADM (see Chapter Five, Sub-sections 5.2 and 5.3).

Present research constraints may be attributed generally to the project time-frame and the requirements of the research sponsor. Significant constraints comprise the following :

(i) human factors integration should not unduly disrupt current JSD practices. Specifically, the integrity of JSD should be preserved and its notations should be exploited for human factors descriptions (see (f) above). It should be noted that these constraints need not necessarily affect the research adversely. On the contrary, the integration of a human factors method around an essentially unchanged JSD method implies that human factors inputs are located at design reference points and practices which are familiar to software engineers. The assimilation of human factors inputs is thus facilitated. Improved uptake is therefore expected to accrue;
(ii) the scope of the current research is limited to the derivation of an integrated method. Specifically, the research scope is confined to case-study demonstrations of the method as opposed to its validation in the field. The research is thus focused on establishing the viability and capability of the method as opposed to validating its efficacy for ensuring design artifacts of superior usability and functionality.[1] Again, it does not necessarily imply that the research would be affected adversely. The reasons are as follows.

Firstly, a superior design artifact could be expected by improving the uptake of human factors inputs. Such improvements are supported by the integrated method via its orderly and inter-related design processes.

---

[1] Direct validation of method efficacy would require exhaustive field studies that involve controlling for design team composition and interactions; the competence of individual designers; characteristics of the design problem; etc. These controls demand resources that could not be accommodated by the present research, i.e. the project resources and time-frame are already taxed heavily by method derivation and demonstration concerns. Thus, method validation had to be deferred to future research.

Secondly, by recruiting established research and design practices to the construction of an integrated method, it would be reasonable to expect that the method would support the specification of better design artefacts (see Chapter Three, Sub-section 3.2). Further implications of these constraints are discussed in Chapter Twelve;

(iii) the scope of the current research (and hence the integrated method) is limited to primary human factors design concerns. Thus, some design concerns will be referenced but not addressed in detail, e.g. organisational and job design; training and personnel selection; and late evaluation (already well-established). As dictated by the project time-frame, the research priority is a breadth-wise integration of human factors and JSD (spanning requirements definition and display design), followed by a depth-wise account of primary human factors design concerns;

(iv) the human factors component of the integrated method (i.e. the structured human factors method) should be targeted at a human factors designer with a working knowledge of JSD. This constraint satisfied the requirements of the research sponsor and is consistent with (iii) above.

In the following sub-sections, general research considerations for human factors integration with SADMs (comprising the scope, requirements and plan of research) are instantiated for the present research concerning the JSD method.

6.2. An Overview of the Jackson System Development Method

JSD involves designing the technical aspects of software systems[2] based on an event model. Thus, the method is well suited for designing real-time systems (Renold, 1989).

JSD comprises three main stages, namely Modelling, Network and Implementation Stages (although the original version of JSD is presented in six stages, other

[2] Although JSD does not include specialised techniques such as physical database design and human factors, it highlights where they should be accommodated in the framework of the method.

methodological characteristics remain essentially unchanged). Each of these stages are defined by explicit design activities and products to guide the user of the method. The scope of the stages are as follows :

(a) *Modelling Stage :* the main purpose of this stage is to capture the subject matter of the target system. Thus, an abstract model of the users' world is defined using a set of entities and their actions. A JSD entity must exist in the real world and is either a person, organisation, or object that performs and/or suffers a relevant sequence of actions. A JSD action is an atomic event in the real world (with specific start- and end-points), about which the system must produce or use information. The time ordering of actions of each entity is described using one or more structured diagrams (see Figure 6-1). Each *model* process communicates with its *real world* process via inputs associated with its actions. The model is thus realised in the computer as a set of sequential processes. Since model processes define the range of functions that a system can support, they are differentiated from function processes which are concerned only with the production of system outputs (see (b) below);

(b) *Network Stage :* this stage is concerned with the functional definition of the target system. Specifically, function processes (comprising input sub-systems, information processes and interactive functions) are specified and connected by information flows to model processes defined in the preceding stage. Thus, a network of concurrent function and model processes is derived. The timing of their information flows and outputs, and linkages with external world processes are also defined. Figure 6-2 shows a schematic representation of JSD specifications. At this juncture, it is pertinent to note that in JSD, design specification concerns are separated from implementation concerns (see (c) below);

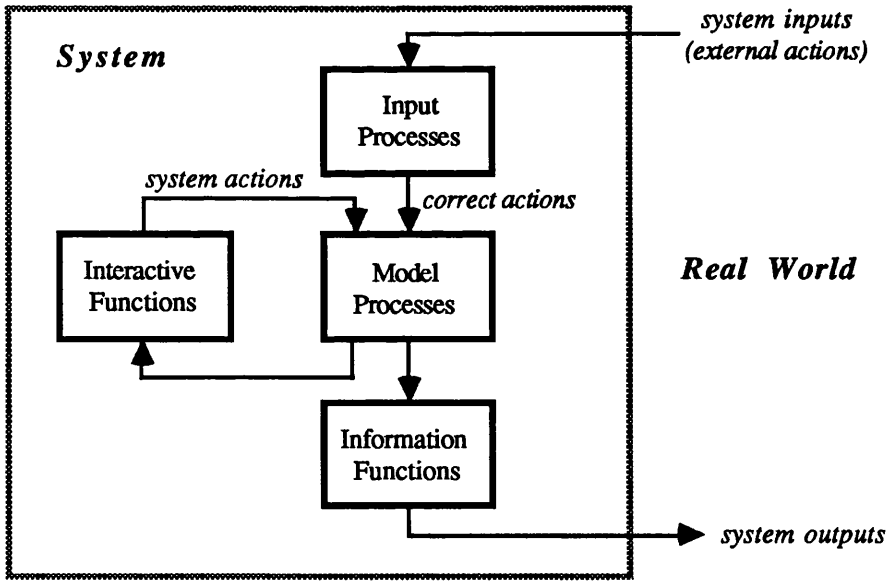
(c) *Implementation Stage :* the purpose of this stage is to transform JSD specifications into sequential programs that may be executed more efficiently by the target hardware. For instance, its concerns include the scheduling of processes, and data storage and access. Specifically, a set of one or more processes are grouped and delegated to a physical or virtual

processor. This transformation of processes would not affect the external behaviour of the system. In other words, the rules of transformation ensure that JSD specifications are preserved during implementation (see Zave, 1984; Renold, 1989). In summary, the characteristics of JSD implementation are as follows :

(1) it is a process that is well regulated (see Zave, 1984) and mechanistic to the extent that most of its transformations could be automated (Renold, 1989). Thus, human factors input during JSD implementation is limited; [3]

(2) it is different from other instances of implementation since it involves a transformation of the specification with respect to the targeted hardware, and not the physical construction of the system.

Figure 6-2 : A Schematic Representation of JSD Specifications (Renold, 1989)



[3] Specifically, human factors inputs are confined to the specification of additional user feedback requirements (e.g. when a particular JSD implementation results in longer than expected transient response time), and an assessment of implications associated with batch and on-line implementation options (e.g. relating user task requirements to computer information updates).

Since software realisation is beyond the scope of legitimate human factors concerns, JSD implementation would not be addressed by the present research, i.e. the research focus is on design specification.

In summary, JSD involves understanding the problem domain before the specification of a design solution. Thus, it emphasises the derivation of a real world model before functional specification, i.e. to define respectively what the system is about before deciding what the system has to do. In other words, system modelling is separated from functional definition. Similarly, design specification is separated from implementation, i.e. what the system has to do is decoupled from how it may be achieved.

This account completes a brief overview of JSD. For a more complete description of the method, the reader is referred to Jackson (1983) and Cameron (1989).

6.3. Identifying the Human Factors Design Support Required by the Jackson System Development Method

Presently, the list of general human factors deficiencies of existing SADMs reported by Anderson (1988) is considered in the context of JSD.

Anderson's observations generally apply to JSD with the following exceptions :

(a) point 4 (Table 6-1) does not apply to JSD since it has been shown to be compatible with object-oriented design approaches (see sub-section 6.1);
(b) point 5 (Table 6-1) may not be entirely true since it was reported that the basic constructs of the JSD structured diagram notation are easily assimilated by users (see sub-section 6.1). However, the use of prototyping should be encouraged for other reasons (see later);
(c) point 6 (Table 6-1) interacts with two demands of a SADM; namely in-depth analysis and specification before implementation; and comprehensive documentation of stage-wise design products. Firstly, it may be argued that the former requirement would reduce the time available for design iterations.

However, this disadvantage should be offset against benefits that accrue from a thorough design analysis phase, e.g. faster convergence to a design solution (see Chapter Two, Sub-section 2.2). Secondly, the emphasis on comprehensive design documentation may encourage a reluctance towards design modifications and iterations. The assertion is based on observations made during the initial introduction of SADMs. With the emergence of CASE tools (e.g. PDF™ and SpeedBuilder™ for the JSD method) the assertion no longer applies.

Table 6-1 : Human Factors Deficiencies of Existing Structured Analysis and Design Methods (after Anderson, 1988)

1. SADMs may not provide procedures for eliciting the operational requirements of the system, e.g. information on users roles, tasks, etc.

2.SADMs may not support function allocation between user and computer. The tendency is to computerise functions that can be automated and leave the remainder to the user.

3. SADMs provide no indication of how the coherence of specifications (e.g. appropriateness of proposed computerised functions) and their implementation may be assessed with respect to the user's task.

4. Some SADMs do not support object-oriented user interface design.

5. The notation of SADMs may be too difficult. Thus, in the absence of prototyping, user feedback elicitation may be supported poorly.

6. SADMs may not encourage iterative design.

7. The scope of SADMs does not adequately address the human-computer dialogue.

8. Physical dialogue design is addressed poorly by SADMs, e.g. there is no reference to style kits and screen design is left to common sense.

Granted these exceptions, human factors deficiencies of the JSD method may be inferred from the remainder of Anderson's list and the human factors design review reported in Chapter Four. Thus, it was concluded that JSD is essentially deficient in two areas of human factors design, namely requirements and task analysis; and user

interface design. Similar conclusions were also reported by Finkelstein and Potts (1985); McNeile (1986); Carver, Clenshaw et al (1987); Carver and Cameron (1987); Carver (1988); Sutcliffe (1988b, 1989); Renold (1989); and Sutcliffe and Wang (1991).

These deficiencies of JSD thus define the scope of human factors design that should be addressed by the present research.[4] A suitable structured human factors method may then be developed for integration with JSD. The requirements corresponding to the present research scope is described in the next sub-section.

6.4. Research Requirements for Human Factors Integration with the Jackson System Development Method

It was stated in earlier Chapters that the scope of the present research comprises the integration of human factors design with a chosen SADM. Thus, human factors integration would involve an *extension* of the design scope of the SADM, e.g. to encompass user interface design. The undertaking should be contrasted against attempts at *enhancing* the practice of the SADM.

Consequently, in the context of JSD, the present research is not concerned with making JSD more usable (e.g. by *enhancing* existing JSD procedures for deriving model and function processes[5]), but with *extending* JSD to include human factors design concerns. Thus, a structured human factors method is developed and integrated with the JSD method. Specifically, the deficiencies of JSD are rectified by incorporating requirements analysis, task analysis and user interface design in the method. The importance of these rectifications are highlighted aptly by the

---

[4] As stated in sub-section 6.1, some human factors concerns would be excluded from the present research scope due to project constraints, e.g. training and personnel selection.

[5] Efforts in this direction have been made by Carver, Clenshaw et al (1987) (see Chapter Five); and Renold (1989).

following statements :

> *"Interfaces cannot be developed as 'add-on' parts of an interactive system, with their development carried out in isolation from the development of the rest of the application system. Thus, an important concept.....is a wholistic approach to interactive system development. Such an approach provides a comprehensive methodology for software design, emphasizing interface development as an integral and equal part of the process.....A wholistic system development enlarges the definition of a 'system' to include both humans and computers.....existing approaches to interface design.....are often ad hoc, unstructured and without cohesion.....results in user interface specifications which appear to comprise random, unconnected messages and displays, and user actions."*
>
> Hartson and Hix (1988)

> *".....it is important to see this task (interface design) in the wider context of system development -- it cannot take place successfully in a vacuum....."*
>
> Newman (1988)

To this end, it was stated earlier that the structured human factors method should satisfy the following requirements :

> (a) it should facilitate the design of a superior design artifact. In the case of a structured method, a superior design artifact is ensured by encouraging an orderly design process via a set of well defined design deliverables. Thus, this requirement is subsumed by requirement (b) below;
> (b) it should support the needs of human factors designers for user interface description and specification. Its human factors design concerns should also be sufficiently explicit to inform software engineers and users. These requirements entail the specification of appropriate stage-wise design scope, process and notation for the method. A review of corresponding requirements implicated by these methodological concerns follows.

Firstly, the *design scope* of the structured human factors method should meet the following requirements :

> (a) it should account for human factors design concerns during system

specification, i.e. appropriate consideration of environment, device, task, and user characteristics. The manner in which such concerns are addressed during system design should be examined to identify stage-wise intersections between the scope of human factors design and the JSD method (see Chapter 4). Design inter-dependencies are similarly identified to support the specification of contact points and information exchanges;

(b) it should compensate for the incompleteness of current human factors knowledge, e.g. by emphasising prototyping and user testing as necessary activities of the method. Thus, prototyping may be necessary to validate design assumptions, etc. As the structured method involves the derivation of stage-wise design products, prototyping and user testing activities could be undertaken at various stages of design.

Secondly, the *design process* of the structured human factors method should meet the following requirements :

(a) it should uphold design principles embedded in JSD, namely modelling before functional design, and the separation of design concerns, e.g. separating specification from implementation. These principles may be exemplified by setting requirements analysis and task modelling before function allocation, and the definition of separate streams for Human Factors and Software Engineering design;

(b) it should facilitate early and continued human factors involvement throughout system design, i.e. the stages of the method should support the derivation of a functional and usable design;

(c) it should support various design scenarios, e.g. variant design, novel design and the computerisation of manual systems;

(d) its procedures need to be comprehensive enough for application by human factors designers with a working knowledge of JSD. However, they should not be too rigid that design creativity is stifled. Appropriate existing design techniques should also be recruited.

Thirdly, the *design notation* of the structured human factors method should meet

the following requirements :

(a) it should be sufficiently powerful to describe the subject matter and rationale of each design stage. In this respect, Frohlich and Luff (1989) suggested that the information to be described during conceptual design should include the following :

(i) the design problem as revealed by requirements and task analysis;
(ii) promising design solutions;
(iii) the implications of each solution with respect to the user's model of the system.

(b) its documentation schemes should adequately record stage-wise design outputs entailed by the method. The records should support an assessment of alternative design solutions;
(c) it should recruit existing JSD notations as much as possible. In addition, although existing notations may be extended or modified for human factors use, the change should be minimal. In this way, the benefits of a common notation may be realised.

The research agenda is thus determined by these methodological requirements. Work activities implicated by the agenda are reported in Chapter Seven.

6.5. An Instantiation of the General Research Plan for Integrating Human Factors with the Jackson System Development Method

In Chapter Three and the preceding sub-section, the requirements for human factors integration with SADMs were defined, namely :

(a) the stage-wise design scope and process of the SADM and structured human factors method should be inter-woven appropriately;
(b) obligatory design inter-dependencies between inter-disciplinary design

streams should be specified;

(c) existing SADM notations should be exploited for describing human factors design products.

It follows from these requirements that appropriate consideration should be given to the characteristics of the chosen SADM during the construction of a structured human factors method. Research activities entailed by these requirements are presently exemplified for the JSD method.

At a high level, the development and integration of a structured human factors method with JSD may be conceptualised as comprising the following sequential activities[6]:

(a) define a set of design stages and products for a structured human factors method that augments the deficiencies of the JSD method (identified in sub-section 6.3). The scope of human factors support is thus constrained;

(b1) define stage-wise design processes and procedures for deriving products of successive design stages specified in (a) above.

(b2) extend JSD notations to include the description of human factors design products specified in (a) above. Since structured design, by definition, implies design derivation and description via a set of stage-wise processes and products respectively, the outputs of (a), (b1) and (b2) constitute a structured human factors method (henceforth referred to as JSD*(HF));

(c) define obligatory contact points between JSD*(HF) and JSD methods; i.e. design inter-dependencies are specified. The two component methods are thus integrated. The JSD and integrated methods are henceforth referred to respectively as JSD* and JSD*(SE) (to indicate that additional design inter-dependencies with JSD*(HF) have been introduced to existing JSD).

At this juncture, specific research constraints may be introduced (e.g. to match case-study tests appropriately to available resources for research), and the activities

[6] Including iterations within (a) to (c) as necessary.

are then operationalised into a research plan. For instance, it can be seen from Figure 6-3 that the current research plan essentially comprises cycles of method specification followed by implementation in case-study tests. A more detailed account of the plan follows.

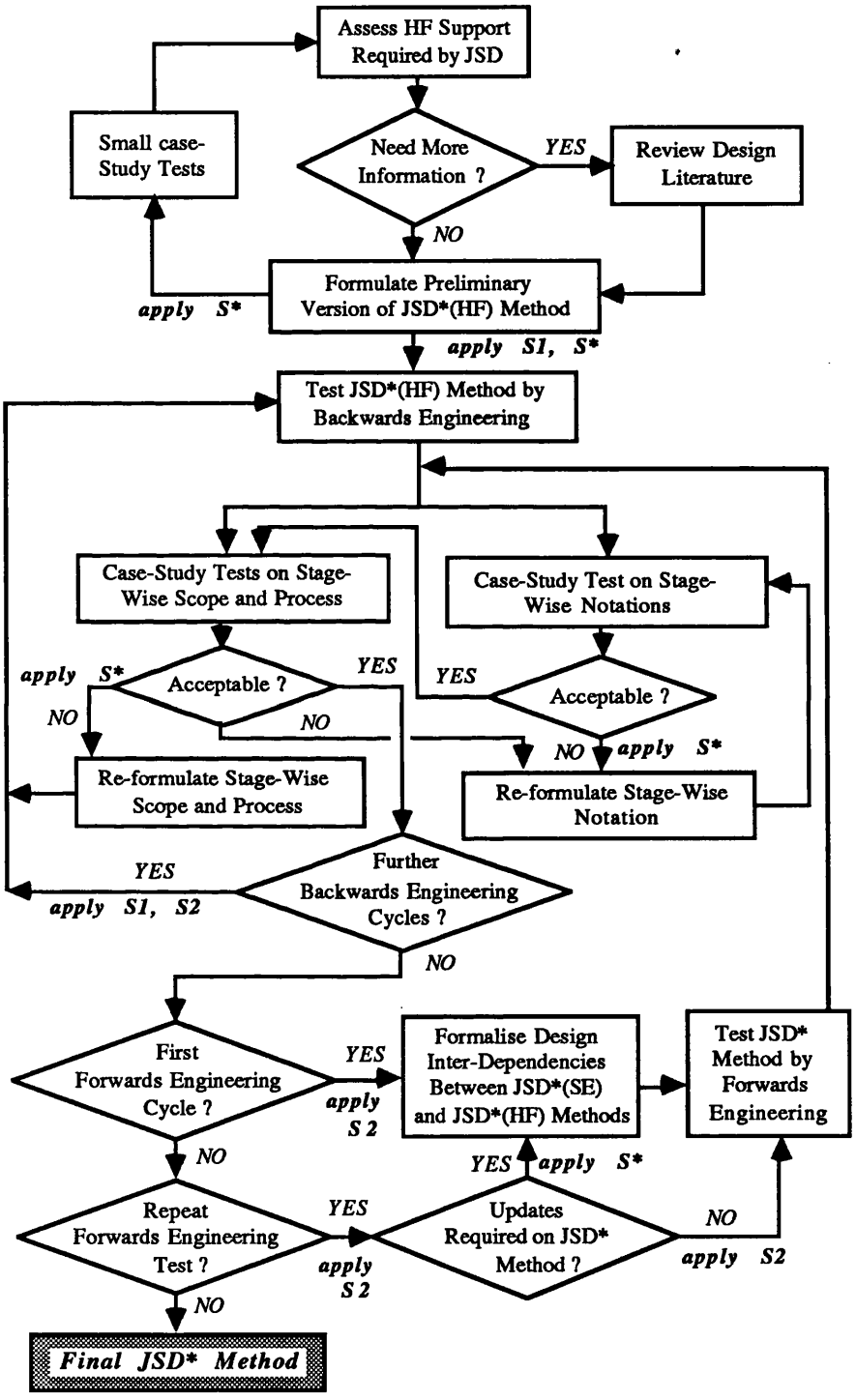
As indicated in (a) to (b2) above, the research is initiated by the specification of a preliminary version of the JSD*(HF) method. The method may be constructed by assessing the stage-wise scope, process and notation of the JSD method to determine the human factors design support required (see sub-section 6.3). The assessment should also expose the potential for recruiting particular aspects of the method, e.g. exploiting JSD structured diagram notation for task description (see Chapter Five, sub-section 5.3.2). On the basis of the assessments, a preliminary version of the JSD*(HF) method (and hence an early version of JSD*) may be specified in two ways. Firstly, a version may be constructed *independently* from existing human factors methods. The efficacy of the derived JSD*(HF) method in complementing the JSD method may then be validated in the field.

Secondly, existing human factors methods and design practices may be surveyed and then recruited to the construction of a JSD*(HF) method. This approach offers greater assurances that the resulting method would support the specification of a superior design artefact, i.e. substantiated conceptions (in the form of established methods and design practices) is more likely to have conceptual and ecological validity.[7] Since the approach builds upon existing knowledge, the following additional benefits may accrue :

(a) quicker convergence to an acceptable JSD*(HF) method. Thus, less iterative testing of method versions is necessary. The early derivation of a JSD*(HF) method would permit more case-study tests to be undertaken

[7] This assertion should be tempered by other factors, e.g. the composition and abilities of design team members. In other words, it is acknowledged that a method alone cannot guarantee superior design. Thus, the efficacy of a method needs to be validated in the field regardless of the research approach adopted.

Figure 6-3 : An Instantiation of the General Research Plan for Human Factors Integration with the JSD Method



Note

S*, S1 & S2 are different research strategies.

S*: Super-ordinate strategy of 'specify followed by implement or test'.

S1: Strategy of 'backwards before forwards engineering'.

S2: Strategy of 'test using case-studies of increasing size and complexity'.

within the project time-frame. Consequently, the derived method is expected to be more robust;

(b) positive transfer of learning in respect of the JSD*(HF) method since users of existing human factors methods would already be familiar with the parts recruited from the latter.

For these reasons, the second research approach is adopted as shown in Figure 6-3. An account of the approach follows.

It was observed in sub-section 6.3 that user interface design is not included explicitly in the JSD method, and should hence be supported by the JSD*(HF) method that is to be developed. Thus, sequential research activities for integrating human factors with the JSD method would comprise the following :

(a) a literature search of existing user interface design techniques;

(b) assessment and recruitment of promising design techniques in (a) above (with modifications and extensions as necessary to meet the requirements of a structured method) to support the construction of a structured user interface design technique for the JSD*(HF) method;

(c) assessment and exploitation of JSD notations (with modifications and extensions as necessary) for describing stage-wise products of structured user interface design in the JSD*(HF) method.

The above sequence of activities is repeated for all areas of human factors design support required by the JSD method, e.g. task analysis. A reasonably complete version of the JSD*(HF) method is thus derived incrementally and subjected to case-study tests. Specifically, parallel case-study design tests are conducted to assess the utility and capability of the following methodological characteristics of the JSD*(HF) method :

(a) notations and documentation schemes for describing stage-wise human factors design products;

(b) design processes for supporting the derivation and transformation of

stage-wise human factors design products. By adopting simple case-study systems (i.e. systems with well defined start- and end-points) for initial design tests, 'missing' design stages may be backwards engineered (see later), e.g. to ensure that JSD*(HF) design transformations are manageable.

In this way, progressively improved and detailed versions of the *JSD*(HF)* method are constructed and tested iteratively on further case-study systems. When a satisfactory version of the JSD*(HF) method is derived, forwards engineering tests may be applied. Following an acceptable outcome of the tests, the integration of JSD*(HF) and JSD methods is then made explicit as follows :

(a) stage-wise concerns of the JSD*(HF) method are inter-woven with appropriate design stages of the JSD method;
(b) obligatory design inter-dependencies[8] between JSD*(HF) and JSD*(SE) methods are specified, e.g. obligatory information exchanges.

Thus, a preliminary version of the *JSD** method is derived. The relationship between Software Engineering and Human Factors design entailed by the JSD* method is then subjected to further iterations of forwards engineering tests. Questions of scale are also addressed at this stage of the research by using larger case-study systems in the design tests. An acceptable JSD* method is thus derived.

At this juncture, case-study assessments that support method development should be elaborated. Generally, assessments on the adequacy of a method comprise subjective judgements in respect of the following :

(a) its support for design derivation, i.e. whether the stage-wise scope and process of the method represent manageable steps for design specification.

---

[8] Situation-specific and informal exchanges between design stages of the component methods need not be described explicitly, i.e. only obligatory contacts points should be made explicit. This assertion is consistent with the requirement that a method should be generally applicable across design domains and project circumstances.

Specifically, the assessment should determine whether :

(i) the scope of the proposed method is reasonably complete. For instance, its design support should span requirements to specification;
(ii) activities within and between design stages are manageably and sensibly organised to support design reasoning. Specifically, design perspectives which vary with the system design cycle should be supported adequately by the *stage-wise scope* and *process* of the proposed method, i.e. assessment of the products and procedures of the method; the coherence of their grouping into design stages, and the sequencing of the stages.

(b) its support for design description, specification and communication, i.e. whether the proposed notations and documentation schemes are sufficiently specific and comprehensive to facilitate inter-designer and designer-user discussions. In particular, the proposed notations should be :

(i) sufficiently powerful to support the description of both intermediate and final human factors design products;
(ii) sufficiently specific to support unambiguous human factors design specification.

Similarly, the proposed documentation schemes should ensure an adequate record of major design decisions and rationale for all stage-wise design products of the method.

(c) its support for design co-ordination and management between JSD*(HF) and JSD*(SE) components of the JSD* method, i.e. whether specified design inter-dependencies are adequate to ensure efficient design.

On the basis of these assessments, appropriate upgrades to a particular version of the method may be inferred. Thus, progressively refined versions of the method are derived, culminating in an acceptable version of the JSD* method.

To address systematically the above concerns during the construction of JSD*(HF) and JSD* methods, particular research strategies were suggested in Chapter Three (sub-section 3.3). Presently, three research strategies applied during the integration of human factors with the JSD method are described.

The super-ordinate strategy (S*) comprises the iteration of method specification followed by its implementation in case-study tests. The version of JSD*(HF) is upgraded between successive specification-and-implementation cycles (see Figure 6-3). Interactions between this and other strategies are described later.

A second research strategy (sub-ordinate to S*) is backwards engineering[9] before forwards engineering (S1). The application of this strategy is shown schematically in Figure 6-3. Essentially, the strategy involves specifying the derivative design capabilities of a method via a sequence comprising backwards and forwards engineering processes (see Chapter Three, sub-section 3.3). Thus, the research is focused initially on backwards engineering the stage-wise products, processes and notations of the JSD*(HF) method.

Following the derivation of a preliminary version of the method, its notations and design processes may be developed concurrently if desired (see Figure 6-3). For instance, notational development could begin with an assessment of the capability of existing JSD notations for describing stage-wise design products entailed by the JSD*(HF) method. Identified notational deficiencies may be rectified by recruiting suitable notations from existing human factors methods. In particular, the notations of methods that have already been ported to the construction of the JSD*(HF) method should be examined and extended if necessary. To ascertain whether the deficiencies observed previously have been rectified, the descriptive capability of the notations is re-assessed using the same case-study system. Following a positive assessment, further notational tests on other case-study

---

[9] Reverse or backwards engineering is used widely as a design recovery technique to decipher the design process for a *finished* product, i.e. to derive a post-hoc understanding of how a particular artefact is designed. The backwards engineered design process is then tested under forwards engineering ('normal' design) to assess its 'goodness of fit' (see Chikofsky and Cross II, 1990).

systems may then be introduced.

A parallel iteration of similar specification-and-assessment procedures applies for developing the design process of the JSD*(HF) method. When a reasonable version of the method is derived, forwards engineering tests may then be applied to assess its capability for supporting design derivation.

It should be noted, at this juncture, that strategy S1 is facilitated by a further strategy (namely strategy S2). The latter strategy involves testing various versions of the JSD*(HF) method using case-study systems of increasing size and complexity. In other words, strategy S2 imposes a range of test demands on successive versions of the method derived using strategy S1. For instance, small and simple case-study systems would be prescribed by strategy S2 at earlier stages of JSD*(HF) development; and on initial application of forward engineering tests on JSD*(HF) and JSD* methods. These systems would be suitable test-beds because :

> (a) at early stages of JSD*(HF) development, the well defined design start-point (design requirements) and end-point (design specification) of such systems would facilitate the construction of a preliminary version of the method. Specifically, the scope, process and notation of intervening design stages may be backwards engineered more easily;
> (b) during the initial application of forward engineering tests, simple systems permit the scale and complexity of the case-study domain to be decoupled from the concerns of method development. In this way, the complexities of the research may be managed better, e.g. method tests using case-study systems of greater size and complexity are introduced only after an acceptable version of the method is derived.

Thus, strategy S2 generally facilitates method development by separating issues concerning method development from those concerning its capability for supporting various design scenarios. The strategy involves addressing the latter set of issues incrementally at major stages of method development, e.g. when moving from

backwards to forwards engineering tests and from JSD*(HF) to JSD*.

In summary, the scope of strategy S1 includes the derivation and testing of various versions of the JSD*(HF) method. Initial versions of the method are generally constructed using backwards engineering. Forwards engineering tests are then introduced when a satisfactory version of the method is derived. Strategy S2 supports strategy S1 by prescribing simple case-study systems as test-beds during the initial stages of method development. The use of larger and more complex case-study systems are deferred until a satisfactory version of the method is derived. Progressively advanced versions of the method are thus specified and tested cyclically in accordance with the super-ordinate strategy, namely S*. The latter strategy prescribes iterative cycles of method specification followed by their implementation in case-study tests. On deriving an acceptable JSD*(HF) method, design inter-dependencies with the JSD*(SE) method are then made explicit to complete the integration of Human Factors and Software Engineering methods. The resulting product constitutes the first version of the integrated method, i.e. JSD*. Further forwards engineering tests are then applied on the integrated method in accordance with strategy S1. During this phase of JSD* development, strategies S2 and S* are applied as before. Progressively advanced versions of the JSD* method are thus generated and assessed, culminating in an acceptable version of the method.

The above account completes an instantiation of the general research plan (proposed in Chapter Three) with respect to the integration of human factors with the JSD method. Subsequently, the instantiated research plan was implemented to support the specification of JSD*(HF) and JSD* methods. The milestones of the implemented plan are reviewed in the following chapter.

Chapter Seven : Research Highlights in Integrating Human Factors with the Jackson System Development Method

"Knowledge advances by steps, and not by leaps."
*Lord Macaulay, 1828, **Edinburgh Review**.*

This chapter provides a historical account of the research activities undertaken for the purpose of specifying and subsequently integrating a structured human factors method with the JSD method. Since the project is extensive (43 reports were written -- see Part VII for a list of working documents), it would be inappropriate here for a complete account of the research to be reported. Consequently, an appropriate selection leading to the final product of the research (i.e. the JSD* method) is now presented.

7.1. An Overview of the Research into Human Factors Integration with the Jackson System Development Method

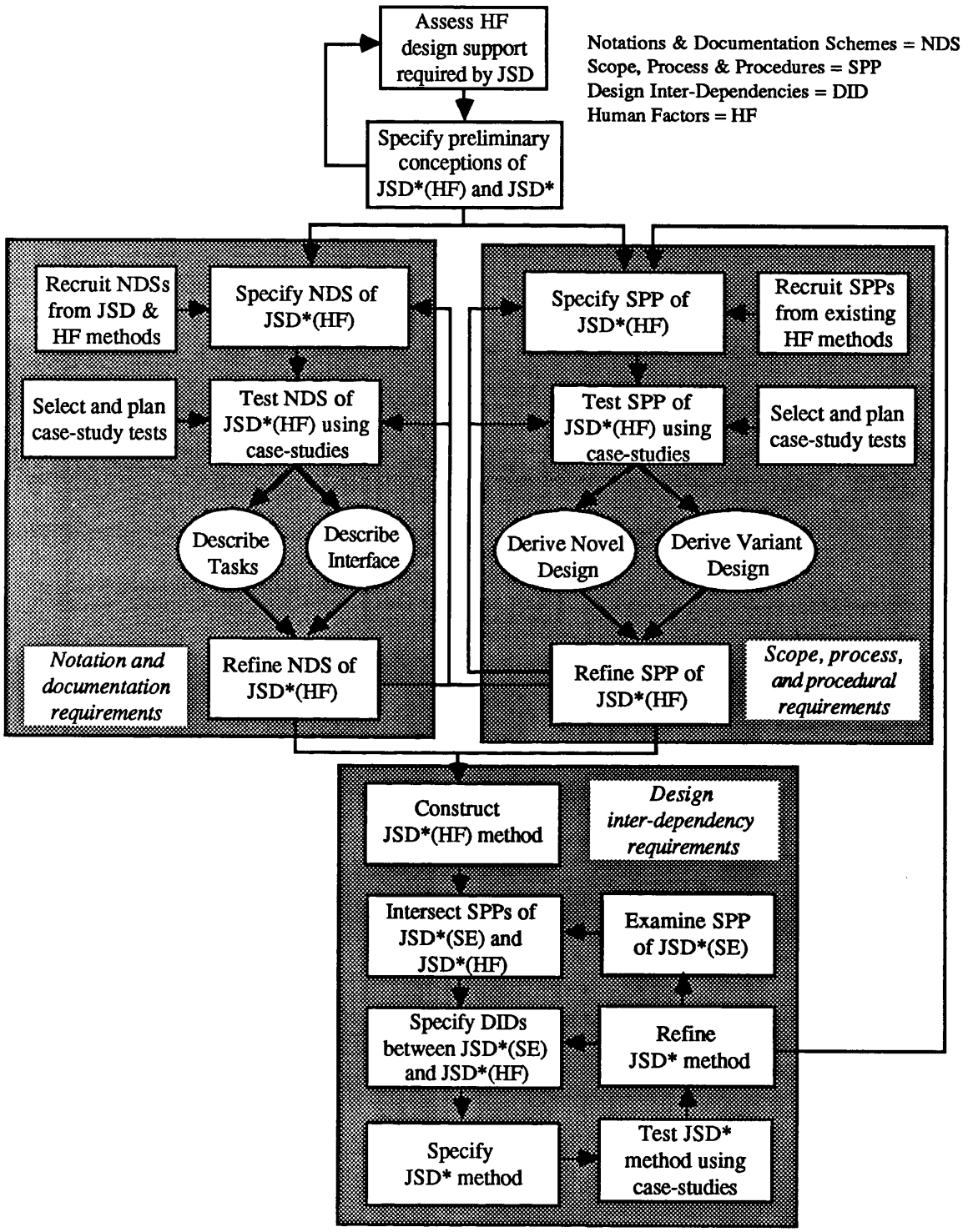
Figure 7-1 summarises the activities of the present research. These activities derive from the research plan described in Sub-section 6.5 (Chapter Six). It can be seen from the Figure that following the specification of preliminary JSD*(HF) and JSD* conceptions, three sets of research requirements (see Chapter Six) were addressed, namely the specification of :

(a) stage-wise notational and documentation schemes;
(b) stage-wise scope, process and procedures;
(c) design inter-dependencies between JSD*(SE) and JSD*(HF) components of the JSD* method.

These requirements were investigated over several research cycles involving various case-study applications (see Table 7-1). Table 7-2 is a chronological profile of the significant developments and reports of the research. A review of highlights

that shaped the research follows.

Figure 7-1 : A Summary of Research Activities for Human Factors Integration with the Jackson System Development Method



7.1.1. A Review of Activities for Specifying Stage-wise Design Notations and Documentation Schemes of the JSD*(HF) Method

The first set of research requirements involved specifying and testing notations and documentation schemes for the JSD*(HF) method. These requirements were *decoupled* initially from those concerning the scope, process and procedures of a method. Such a research scheme was supported by the recruitment of existing JSD notations. In addition, the research scheme was facilitated by an early specification of preliminary JSD*(HF) and JSD* conceptions, which was supported in turn by the well reported human factors design requirements of the JSD method (see Chapter Five, Sub-section 5.3). Consequently, the notational concerns at this stage

Table 7-1 : Profile of Case-Study Applications Used in the Research

Year	Case-Study Applications
1987-1988	MacDraw™ (Macintosh-based)
	DisplayWrite™ (PC-based)
1988-1989	MacDraw™
	Automatic Teller Machine
	Library Management System
	London Transport Ticketing Machine
	WriteNow™ menus
	IKBS Troubleshooting System
	Digital Network Management System Simulator
	Digital Network Management System
	Recreation Facility Booking System
	MicroSoft Disk Operating System™
1989-1990	*Recreation Facility Booking System*
	Digital Network Management System
	Home Energy Management System Prototype

Note : Applications highlighted in italics were case-studies used extensively for method development. Other applications were recruited selectively to progress parts of the method.

Table 7-2 : Profile of Research Developments and Reports

Year	Research Developments	Working Document No.
1987-1988	** *Setting project aims and objectives.*	*WDs 1, 2.*
	* * Brief review of Software Engineering methods.	WDs 3, 4.
	* * Initial examination of JSD structured diagram notation and design process.	WDs 5, 6, 7, 8.
	* * Initial study of existing human factors contributions to user interface design.	WDs 9, 10, 12, 13.
	** *Defining methodological integration and design terminologies.*	*WD 11.*
	* * Initial framework for achieving methodological integration.	WD 14.
	** *Generating an initial conception of human factors integration with JSD.*	*WDs 15,* 17,*19,* 20.
	** *Specifying case-study plans for the research.*	*WDs* 16,*18, 22, 25.*
1988-1989	* * Initial work on the Digital Network Management case-study.	WD 21.
	** *Specifying JSD* terminology, stage-wise scope, process and notation.*	*WDs* 23, 24,*26, 27.*
1989-1990	** *Generating a detailed conception of JSD*.*	*WD 28.*
	** *Testing JSD* using the Recreation Facility Booking System case-study.*	*WD 29.*
	* * Further recruitment of human factors design techniques to JSD*.	WDs 30, 31, 32.
	** *Specifying the first version of the JSD* method.*	*WDs 33, 34.*
	** *Specifying research plans for the Digital Network Management case-study.*	*WD 35.*
	** *Testing and generating successive versions of the JSD* method.*	*WDs* 36, *37A,* 37B, *38A,* 38B,*39A,* 39B, *40A.*

Note : Research milestones and major project reports are highlighted in italics.

of the research comprise the following :

(a) *substantiating* reports on the utility of JSD structured diagram notation for human factors description (Chapter Five, Sub-section 5.3). In particular, a range of JSD notations were tested on their capability for describing tasks (at both conceptual and interaction levels -- see Table 7-3) and screen objects (e.g. the behaviour of menus and windows). Case-study applications used for these tests comprise parts of the following : MacDraw™ (Macintosh-based); a Library Management System; an IKBS Troubleshooter (Poltrock et al, 1986); and Automatic Teller Machines (various versions). The tests indicated JSD structured diagram notation to be particularly suitable for describing time-ordered events and object-oriented information. The results thus confirmed earlier reports on the utility of JSD modelling notation (comprising entity and action lists and structured diagram notation). In contrast, the recruitment of its network diagram notation could not be supported unequivocally. A detailed account of these research concerns may be found in Lim (1987; 1988d; 1988e);

Table 7-3 : Information Description Capability of Existing JSD Notations

Information Type	*JSD Entity/Action Lists*	*JSD Structured Diagram Notation*	*JSD Network Diagram Notation*
Semantic	X (e.g. names and descriptions of objects and actions)	X (e.g. names and descriptions of nodes and sub-nodes)	X (e.g. entity relationships)
Syntactic	Debatable (e.g. definition of attributes ?)	X (e.g. JSD structured diagram constructs)	
Lexical	X (e.g. atomic actions)	X (e.g. leaves of JSD structured diagrams)	

(b) *justifying* the choice of JSD structured diagram notation by demonstrating its capability for human factors description to be at least equivalent to other established notations. Thus, comparisons were made with grammar-based notations (e.g. BNF (Reisner, 1977); TAG (Payne and Green, 1986)); networks (e.g. GTN (Kieras and Polson, 1985)); flowcharts (e.g. Drury, 1983) and tree hierarchies (e.g. HTA (Annett and Duncan, 1967). Sample descriptions (e.g. DisplayWrite™ tasks) published by the preceding authors were re-described using JSD structured diagram notation. The objective was to demonstrate that the latter notation is capable of describing the same information with at least the same specificity and 'elegance' (e.g. conciseness, modifiability, learnability). In addition, reported weaknesses of the notations were noted. The results of these comparisons were favourable (see Walsh, 1987a and b).

Following the decision to use the JSD structured diagram notation, the research focus shifted onto the scope, process and procedures of a method. In other words, subsequent notational developments were contingent on the identification of additional requirements to support the design task, e.g. the description of intermediate design representations to support reasoning. Similarly, documentation schemes were developed to ensure a comprehensive record of design decisions and design rationale. Thus, notational and documentation concerns pervaded the course of the research. Notable developments in respect of these concerns are as follows :

(a) the inclusion of a hierarchy construct in JSD structured diagram notation. The construct represents a more elegant way of describing non-sequential events, since it replaces the need to use a combination of iteration and selection constructs for such descriptions (see Annex A; Lim, 1988e, 1989b). Thus, the hierarchy construct represents a useful addition to support the use of the notation for task description;

(b) the relaxation of description rules associated with JSD structured diagram notation (Lim, 1988e). Specifically, rules which were intended originally to ensure the direct executability of JSD specifications were relaxed to accommodate the 'fuzziness' of human factors descriptions, for

instance the rule :

(i) 'described actions should be atomic' is interpreted less stringently to imply that the atomicity of actions is dictated by the purpose of the description. Thus, 'chunking' of sub-tasks may be accommodated;
(ii) 'described actions should have detectable start and end points' is not always upheld. Although the rule may be satisfied in most instances, where overt actions are described, compliance with respect to cognitive processes can not be assured. Thus, its application is relaxed.

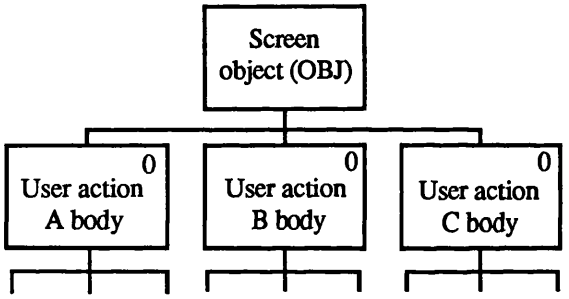
(c) an extended application of JSD structured diagram notation to include the following design descriptions :

(i) spatial layout and composition of screen objects (Lim, 1989b). The objective of using JSD structured diagram notation for such descriptions was to capture the information in machine readable form. This undertaking constituted an opportunistic attempt (as opposed to a formal research goal) to exploit the capabilities of the structured diagram notation. Unfortunately, following case-study description tests (Recreation Facility Booking System and Digital Network Management System Simulator), it was concluded that extensions to notational interpretation, required to widen its applicability, would constitute non-trivial modifications to the notation. Since these modifications might have resulted in negative transfer to JSD analysts, the attempt was abandoned in favour of a simple pictorial description (which provides essentially the same information);
(ii) behaviour and changes in appearance of screen objects. Since the JSD structured diagram notation is particularly suitable for describing objects, only minor extensions were necessary to link the behaviour of screen objects to changes in their appearance. Such descriptions are effected by assigning unique identifiers to object actions so that they may be linked to pictorial descriptions of appearance changes (see

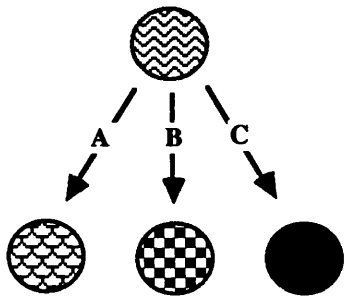
Figure 7-2);

(iii) screen transitions. The transitions correspond to the dynamic presentation of functional supports, and error and feedback messages to the user. The description (termed the Dialogue and Inter-Task Screen Actuation Description (DITaSAD)) is linked to screen layout diagrams and message index tables (Lim, 1989d, 1990d, 1990e). Its purpose is to contextualise major screen presentations to the progress of the user's task.

Figure 7-2 : Using JSD Structured Diagrams to Link Screen Object Behaviours to Corresponding Changes in their Appearance



A, B and C are permissible selections of user actions on an object named OBJ.



Changes in appearance of object OBJ corresponding to actions A, B and C.

(d) a wider recruitment of existing JSD notations, in particular its network diagram notation. Specifically, the potential of the notation for describing information exchanges between system entities was investigated, e.g. the control of information flows, such as its direction of flow and timing. To this end, the notation was considered for the following descriptions :

(i) a conceptual representation of information exchanges between the human, computer and other real world entities, e.g. whether particular information flows and updates are continuous, periodic, or only effected on request;

(ii) the nature of information exchanges between users in the

organisation. Such descriptions would be useful when socio-technical considerations are addressed in system development;

(iii) the display of screen objects, e.g. whether particular functional objects are always presented.

Following case-study tests (Recreation Facility Booking System and Digital Network Management System Simulator), it was concluded that substantial investigations would be required to use the notation for the above descriptions. Although the potential benefits of such recruitment were acknowledged, their investigations would have entailed a re-allocation of project resources and corresponding changes in the research scope. Thus, the investigations could not be supported, since other research commitments assumed greater priority. Consequently, further research in this direction is deferred to follow-up projects;

(e) the investigation and subsequent recruitment of other (non-JSD) notations and documentation schemes, namely :

(i) network-type diagrams which were used to describe relational information such as composition and taxonomic relationships among objects. In particular, the diagram is used to describe design concepts in a representation termed the Domain of Design Discourse (DoDD) description (Lim, 1989d, 1990a). The description is intended as a means for eliciting and communicating the boundaries of the design problem (see Valusek, 1988);

(ii) circuit-type diagrams which were used to describe interaction task pathways and activation schedules of screen objects (Lim, 1989b, 1989d). However, the diagram was dropped for two reasons. Firstly, the notation was found to be unduly cumbersome to generate and to update during a case-study test involving the Recreation Facility Booking System. Secondly, its description of sequential events is insufficiently specific, whenever feedback loops occur. Thus, circuit-type diagrams were replaced with a combination of JSD structured diagrams and pictorial screen layout diagrams. The latter set of

diagrams was used to describe screen transitions corresponding to the presentation of major functional supports for the users' task, and error and feedback messages;

(iii) pictorial screen layout diagrams which were used to describe the location, grouping and appearance of screen objects. The screen diagrams are labelled such that their sequential and inheritance relationships are identifiable (see Lim, 1990d);

(iv) object appearance diagrams which were used to describe changes in screen object appearance resulting from corresponding changes in state, e.g. following permissible actions on the object (see Figure 7-2; Lim, 1990d);

(v) information tables which provide textual descriptions to support : JSD structured diagram descriptions; pictorial screen layout diagrams; object appearance diagrams; and network-type diagrams. For instance, the tables may be used to document the decisions and rationale underlying a particular design (see Chapters Nine to Eleven).

This account completes a profile of the research highlights associated with the development of stage-wise notations and documentation schemes for the method. Case-study illustrations of the final set of JSD*(HF) notations and documentation schemes are presented later in Part IV.

7.1.2. A Review of Activities for Specifying Stage-wise Design Scope and Processes of the JSD*(HF) Method

In Sub-section 7.1, it was proposed that preliminary JSD*(HF) and JSD* conceptions may be specified to meet design support requirements particular to the JSD method (also see Chapter Five, Sub-section 5.3). Since the notation for the JSD*(HF) method was constrained largely to the use of existing JSD notations, a greater proportion of the research effort spent on generating JSD*(HF) conceptions was directed at specifying appropriate stage-wise design scope and processes for the method. The research emphasis also resulted from the implicit and largely incomplete conceptualisation of human factors design. Thus, a number of

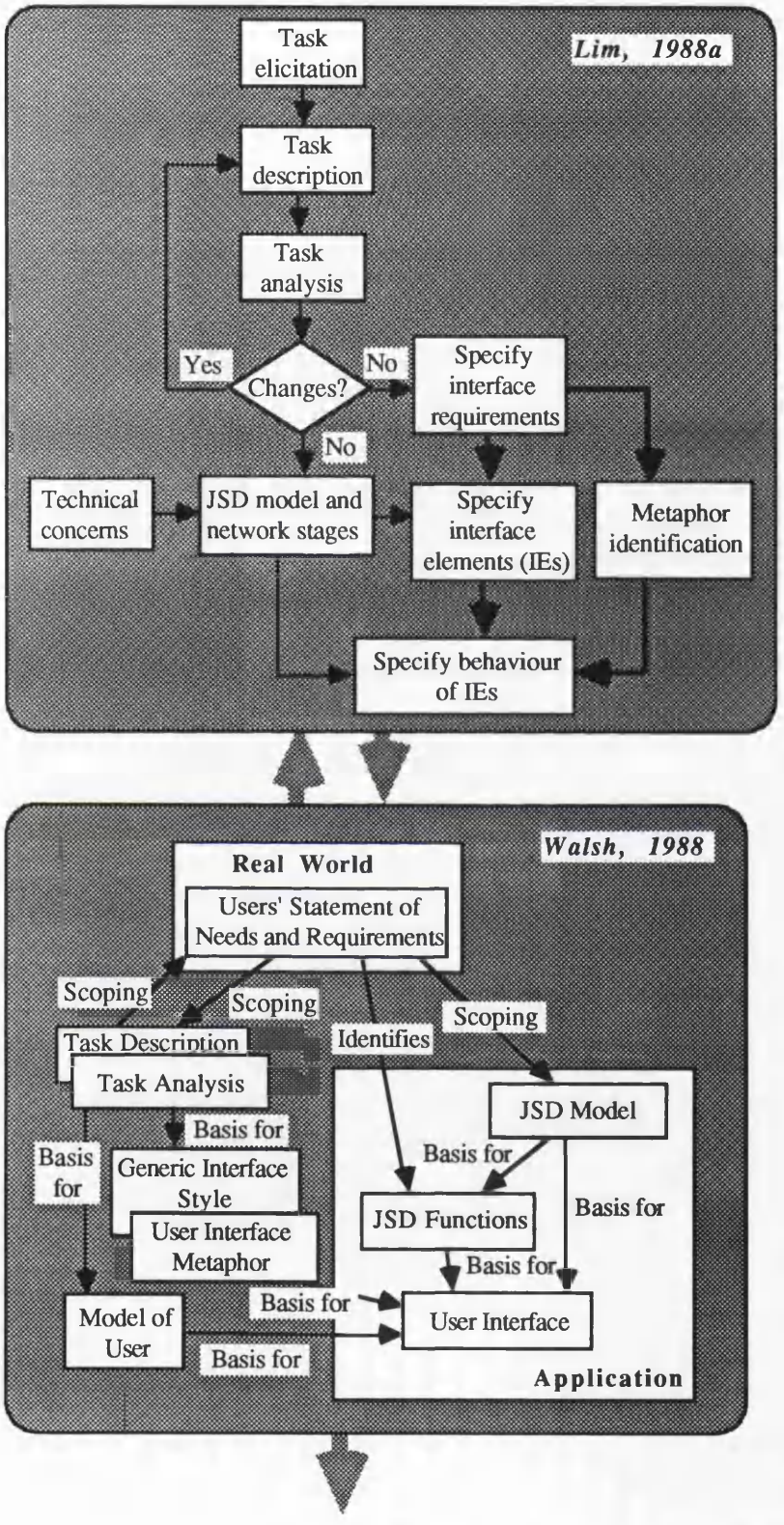
preliminary JSD*(HF) and JSD* conceptions were derived as shown in Figures 7-3 and 7-4.

On the basis of these preliminary conceptions, the methods were then elaborated via appropriate recruitment and extensions of existing human factors techniques. To support the recruitment of existing methods, a general conception of structured human factors design was derived (Chapter Four) and intersected with preliminary JSD*(HF) conceptions. Thus, pertinent aspects of the general design conception were recruited (with some backwards engineering and 'brain-storming') to develop more detailed conceptions of the JSD*(HF) method. A survey of relevant human factors techniques was then initiated. The scope of the survey included system-sub-system definition (which addresses design concerns at the organisational level), user characterisation, task analysis and design, user interface design, performance specification, and function allocation. Although the survey was comprehensive, it was not feasible to undertake an in-depth study of all human factors design concerns in view of the limited research time-frame. Thus, later studies focused directly on addressing design support deficiencies of the JSD method. The final scope of the survey is shown in Figure 7-5 (this may be contrasted with Figure 4-5). An account of the activities and conceptual developments at this stage of the research follows.

The conception shown in Figure 7-5 was detailed further by 'brain-storming' and literature reviews on the following :

(a) general design studies, e.g. Jones (1973); Coyne (1988); Rouse and Boff (eds., 1987); *Design Studies* (Design Research Society Journal, Butterworh Scientific Ltd.); etc. The review afforded an understanding of design tasks and the implications of methodological support, e.g. the impact of methods on design practices (see Part V), and how methods are used by designers (see Part IV on the resulting need for flexibility in a method);
(b) general concepts on human factors design, e.g. the use of metaphors in user interface design. The literature reviewed included texts and papers by Card, Moran and Newell (1983), Rouse and Boff (eds., 1987); Rasmussen (1986), Shneiderman (1987); Helander (ed., 1988); Hutchins (1987);

Figure 7-3 : Initial JSD* and JSD*(HF) Conceptions



Conception in Figure 7-4

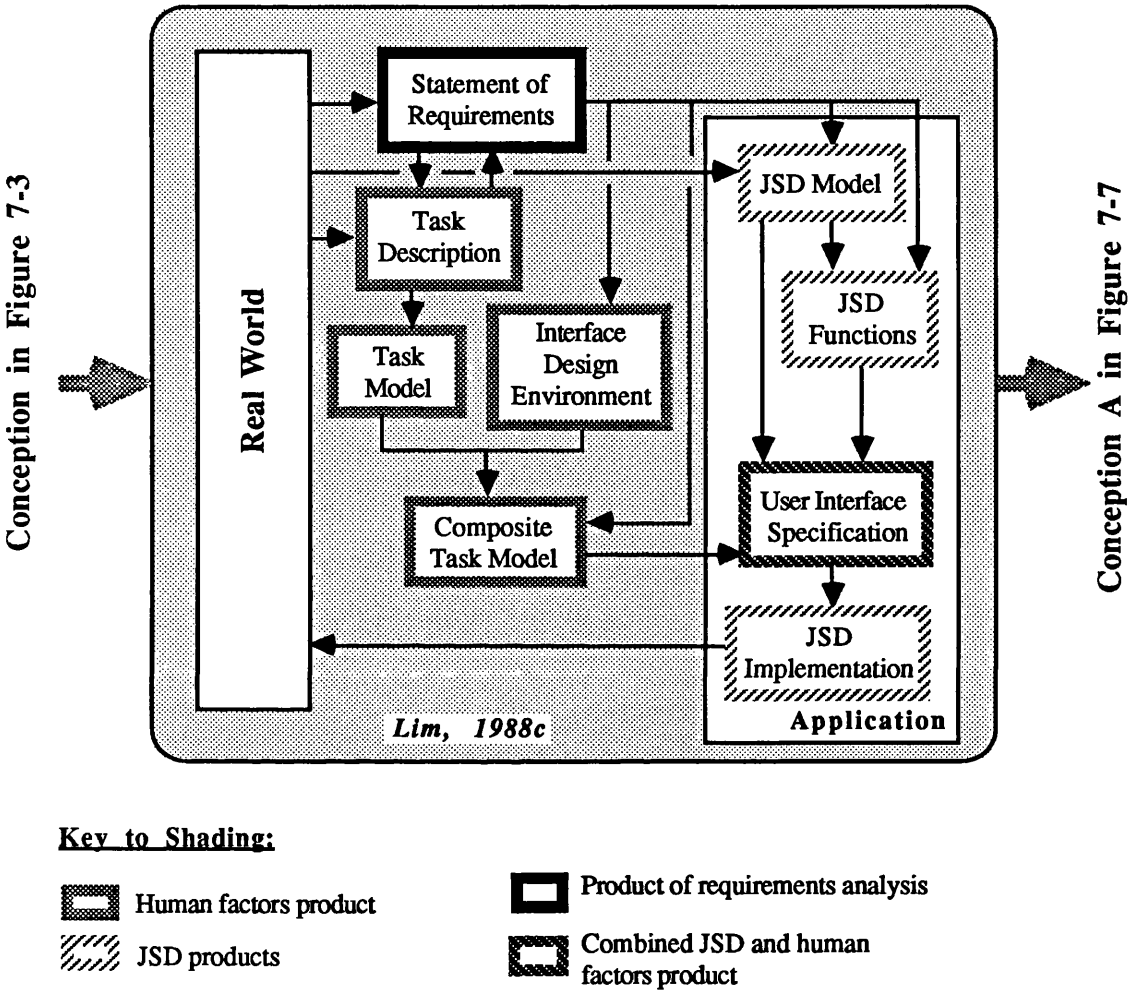
An initial attempt at specifying JSD*(HF) and JSD* conceptions based on observed design support deficiencies in the JSD method. The focus of the conception was on :

(a) incorporating outputs of task analysis into JSD design;
(b) ensuring user interface design and metaphor identification (if any) is based on outputs of task analysis;
(c) specifying two contact points between human factors design and the JSD method;
(d) specifying appropriate design roles. Specifically, user interface design should be undertaken by human factors designers, while technical aspects of system specification should continue as the domain of JSD analysts.

This JSD*(HF) and JSD* conception was derived by a co-worker at about the same time as the one above. The focus of the conception was on :

(a) ensuring task analysis is constrained by user requirements (if well defined);
(b) incorporating outputs from task analysis into user interface design;
(c) deriving an appropriate expression of the user interface based on a model of the user and the target task;
(d) identifying an appropriate user interface style and metaphor identification (if any) based on task analysis;
(e) specifying a contact point between human factors design and the JSD method;
(f) specifying appropriate design roles. Specifically, user interface design should be undertaken by human factors designers, while technical aspects of system specification should continue as the domain of JSD analysts.

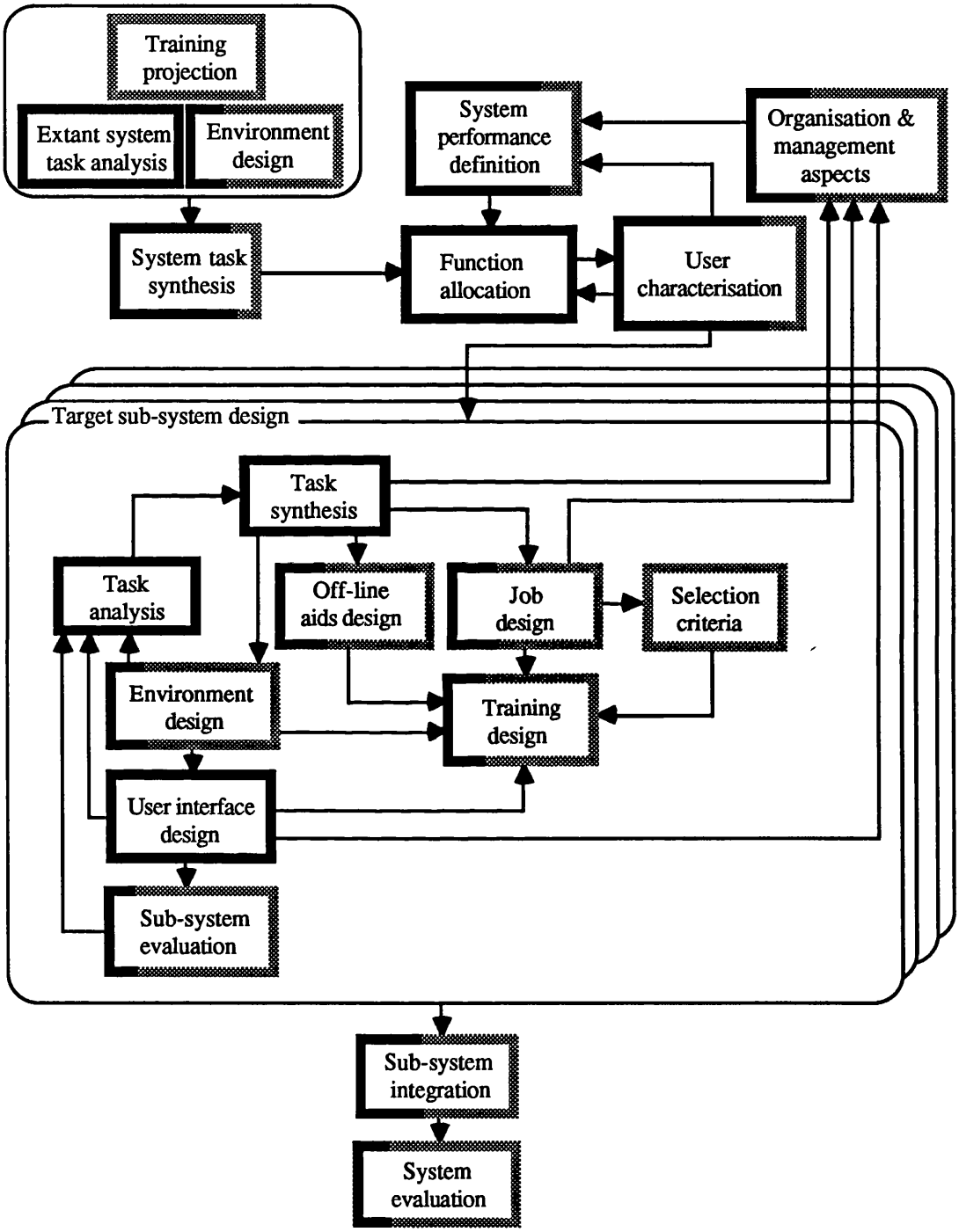
Figure 7-4 : A Product-Oriented Conception of JSD*



The two JSD* and JSD*(HF) conceptions shown in Figure 7-3 were combined. Notable changes comprise the following :

(a) the user model is subsumed by the task model;
(b) human factors and JSD design products are distinguished explicitly;
(c) a composite task model is introduced to address different levels of task design and description.

Figure 7-5 : Scope of the Review of Current Concerns of Human Factors Design



Note :

Bold outlined boxes represent design concerns that are addressed by the structured HF method. Incomplete coverage by the method is represented by boxes partially outlined in bold.

Moran (1981); etc. The objective of the review was to establish a suitable knowledge base to support the specification of appropriate stage-wise design scope and processes for the JSD*(HF) method;
(c) requirements elicitation and task analysis techniques. Following a preliminary review of several task analysis techniques, a set of promising candidates were selected for detailed study. The set included :

(i) Hierarchical Task Analysis (Annett and Duncan (1967) among other reports);
(ii) Task Analysis for Knowledge Descriptions (Johnson et al (1984) among other reports);
(iii) Knowledge Analysis of Tasks (Keane and Johnson (1987) among other reports);
(iv) Hierarchical Planning (Sebillote, 1988);
(v) GOMS (Card et al (1983) among other reports).

For detailed accounts of these reviews, the reader is referred to Silcock (1989), Coles and Lim (1989), and Silcock and Lim (1989). It should be noted that the review of task analysis techniques also included function allocation techniques, e.g. reports by Price (1985) and Clegg et al (1989).

In general, the review indicated that existing task analysis techniques are poorly structured. Specifically, their procedures are frequently incomplete or poorly described; their scope is too narrow (e.g. the applicability of a technique may be limited to evaluation or training only), or too specific for wider application (e.g. GOMS). In addition, some task analysis techniques were still being developed at the time of the review. Nevertheless, the review supported the development of an extant systems task analysis technique for the JSD*(HF) method. For instance, general task analysis concepts and procedures were recruited, e.g. to support target system design, current system information may be abstracted to reveal its logical design (see Johnson and Johnson, 1988) and generified to derive a generalised description;

(d) models of human-computer dialogue and user interface design methods. Both linguistic and non-linguistic models of human-computer dialogue were reviewed. The former model is exemplified by CLG, while non-linguistic models comprise dialogue cell (Borufka et al, 1982); interaction event (Benbasat and Wand, 1984); architectural abstraction models (e.g. Coutaz, 1985, 1989).

An initial review indicated that human factors design is not supported equally well by these models. For instance, architectural models do not appear to consider user needs at all. In this regard, Hartson and Hix (1988) reported that linguistic models are most suitable for incorporating human factors design (see Table 7-4). Thus, linguistic models and their design methods were studied in greater detail. The first linguistic model was proposed by Foley and Wallace in 1974. Since then variants on the basic theme followed, namely linguistic models comprising varying numbers of semantic, syntactic and lexical levels, e.g. Moran (1981), Buxton (1983). Although linguistic models constitute a good scheme for *describing* human-computer dialogue, their emphasis on a rigid top-down description has been found to be inappropriate for supporting user interface *specification* (Sharratt, 1987; 1988). In response to these problems, modifications of the model have been proposed, e.g. Frohlich and Luff (1989) suggested that semantic and syntactic design should be undertaken simultaneously (see also Preece et al, 1987).

Following comparisons between methods based on the linguistic models (e.g. CLG and Foley's user interface design method) and early conceptions of the JSD*(HF) method (see Lim 1989c; Tables 7-5 and 7-6), it was concluded that the latter did not require an explicit incorporation of a linguistic model. Instead, the model's function may be embodied in object oriented descriptions (see Chapter Six, Sub-section 6.1, point (d)). The latter were chosen for the following reasons :

(i) object-oriented descriptions are compatible with JSD*(SE) descriptions generated at the JSD Model Stage;

Table 7-4 : Scope of Description of Various Models of Human-Computer Dialogue (after Hartson and Hix, 1988)

Hartson and Hix's List of 'Essential' User Interface Design Concerns	*Addressed by Methods Based on :*
(a) the use of task analysis to specify a structural description and representation of the interface, i.e. user interface design is driven by task-oriented models (see Kieras and Polson, 1985).	Linguistic Models
(b) the use of structural models to capture the general process of human-computer communication, i.e. describing the structure of inputs and outputs. Dialogue objects and their relationships are thus identified and the domain defined.	Linguistic Models Architectural Abstraction Models (?)
(c) the use of interface models to represent particular instances of human-computer interaction, i.e. specifying the form, content and sequencing for parts of a user interface. Thus, a device-oriented definition is derived.	Linguistic Models Architectural Abstraction Models Dialog Cell Model Interaction Event Model

Table 7-5 : Comparison of User Interface Design Concerns of CLG (Moran, 1981) and the JSD*(HF) Method

User Interface Design Concerns Addressed by CLG	*Corresponding Design Coverage by Design Stages of the JSD*(HF) Method (Approximate)*
Conceptual component	Extant Systems System Analysis to System and User Task Model Stages.
Communication component	Interaction Task Model and Interface Model Stages.
Physical component	Display Design Stage.

Table 7-6 : Comparison of User Interface Design Concerns of Foley and van Dam's Method (1982) and the JSD*(HF) Method

Foley's User Interface Design Method		*JSD*(HF) Design Method*
(1) Specifying the Interaction Concept Stage -- define :		(1) Extant Systems System Analysis to Composite Task Model Stages :
(a) set of all objects in the system	⟷	(a) JSD object/entity lists
(b) relationship between objects	⟷	(b) Domain of Design Discourse
(c) object properties	⟷	(c) JSD object/entity lists
(d) permissible operations on objects	⟷	(d) JSD structured diagram descriptions
Shortcomings (after Frohlich and Luff, 1989) :		Handled by :
(a) recognition of composite and meta-objects	⟷	(a) domain of design discourse desccription
(b) differentiation of action types, e.g. specifying local and global actions.	⟷	(b) constructs of JSD structured diagram notation, e.g. common actions across entity structures may be considered global.
(2) Specifying the Interaction Semantics Stage -- define :		(2) System and User Task Model (SUTaM) to Display Design (DD) Stages :
(a) function name	⟷	(a) SUTaM sub-nodes and JSD function names (?)
(b) action on objects	⟷	(b) SUTaM descriptions
(c) definition of both system and user oriented operations	⟷	(c) SUTaM descriptions
(d) parameters of operations	⟷	(d) JSD action list
(e) results of operations	⟷	(e) JSD functions and DD descriptions
(f) errors	⟷	(f) JSD input sub-system and DD descriptions
(g) feedback messages	⟷	(g) JSD input sub-system and DD descriptions
(h) performance measures	⟷	(h) statement of user needs table
(3) Specifying the Interaction Syntax Stage -- define :	⟷	(3) System and User Task Model (SUTaM) to Display Design (DD) Stages :
(a) sequencing of inputs and outputs (I/Os)	⟷	(a) SUTaM and Interaction Task Model (ITM) descriptions
(b) specification of input and output tokens.	⟷	(b) ITM and Interface Model (IM) descriptions
(4) Specifying the Interaction Lexicon Stage -- define :	⟷	(4) ITM to Display Design (DD) Stages :
(a) mouse movements and key presses	⟷	(a) ITM descriptions
(b) screen layouts	⟷	(b) DD descriptions
Shortcomings (after Frohlich and Luff, 1989) :		Handled by :
(a) outputs would be described better by scenario diagrams	⟷	(a) DD descriptions and JSD*(HF) descriptions in general

(ii) object-oriented descriptions are independent of their implementation, since such information is concealed within object representations. The descriptions can thus accommodate context-free representations of specific interface features, e.g. the behaviour of windows;

(iii) the strong hierarchical inheritance of object-oriented descriptions helps to ensure consistency;

(iv) the class concept of object-oriented descriptions facilitates easy implementation of variant designs. Thus, design re-usability and prototyping is supported.

Nevertheless, a system-oriented design perspective should be maintained as information on temporal sequences (e.g. intermediate execution steps) is obscured inherently by object-oriented views (Hartson and Hix, 1988). Thus, the characteristics of user interface objects and actions should relate to tasks that users are expected to perform.

(e) classes of human factors models and their potential contributions to system design. The review sought to uncover :

(i) the nature, type and expression of human factors models. Many papers were reviewed including those by Clowes (1988), Farooq and Dominick (1988), Johnson (in press), Long (1986, 1987), Nielson (1987), Norman (1983), Simon (1988), Whitefield (1987), Young (1983), etc. The review identified a profusion of human factors models and considerable confusion in their interpretation (Long, 1987). Attempts to resolve this state of affairs via definitions, classifications and taxonomies (e.g. explicit specification of who is doing the modelling and what is being modelled (see Whitefield, 1987)) have successfully clarified the nature and type of human factors models. However, the current representation of models (e.g. 'models of the user') is either left implicit and poorly described, or too theoretical and complex for practical application.[10] Thus, it was

concluded that system and task models constitute more promising recruits to the JSD*(HF) method. Nevertheless, by adopting a user-oriented approach an implicit user model may be subsumed by particular characteristics of system and task models;

(ii) how human factors models were used in design. The review indicated that research into human factors models was still in its early stages (see Footnote 10). Thus, few case-study applications had been reported and the assumed user models tended to be described poorly (see van der Veer et al, 1988; Guest, 1988). The review also indicated that human factors models were used primarily as explanatory aids in diagnostic evaluation, as opposed to predictive aids to support design generation[11] (see Whitefield, 1990);

(iii) how human factors models are transformed between stages of the system design cycle. In this respect, current definitions, taxonomies and classifications were generally concerned with categorising *existing* human factors models (which may not cover system design needs). Such classifications do not adequately support the application of human factors models. In particular, no information is provided on how particular models may be recruited to support design at each stage of the design cycle. Although attempts were made to identify the protagonist of human factors models (e.g. designer's models, user's models, etc.), the subject matter addressed by the models was not intersected explicitly with specific or stage-wise design concerns. Thus, the recruitment of human factors models is usually confined to the late evaluation stage, e.g. to check the designer's interpretation against the user's conceptual model (see also (ii) above on the primary use of human factors models as explanatory aids). Consequently, to facilitate earlier recruitment (e.g. during design specification), human factors models should be set explicitly against specific stages of the system design cycle (or embedded in a structured method). Figure 7-6 shows the result of such an attempt (see Lim (1989a) for further

[10] The situation is probably a result of the incomplete human factors knowledge in this area.

[11] With the possible exception of task models.

Figure 7-6 : Setting Human Factors Models Against the System Design Life-Cycle

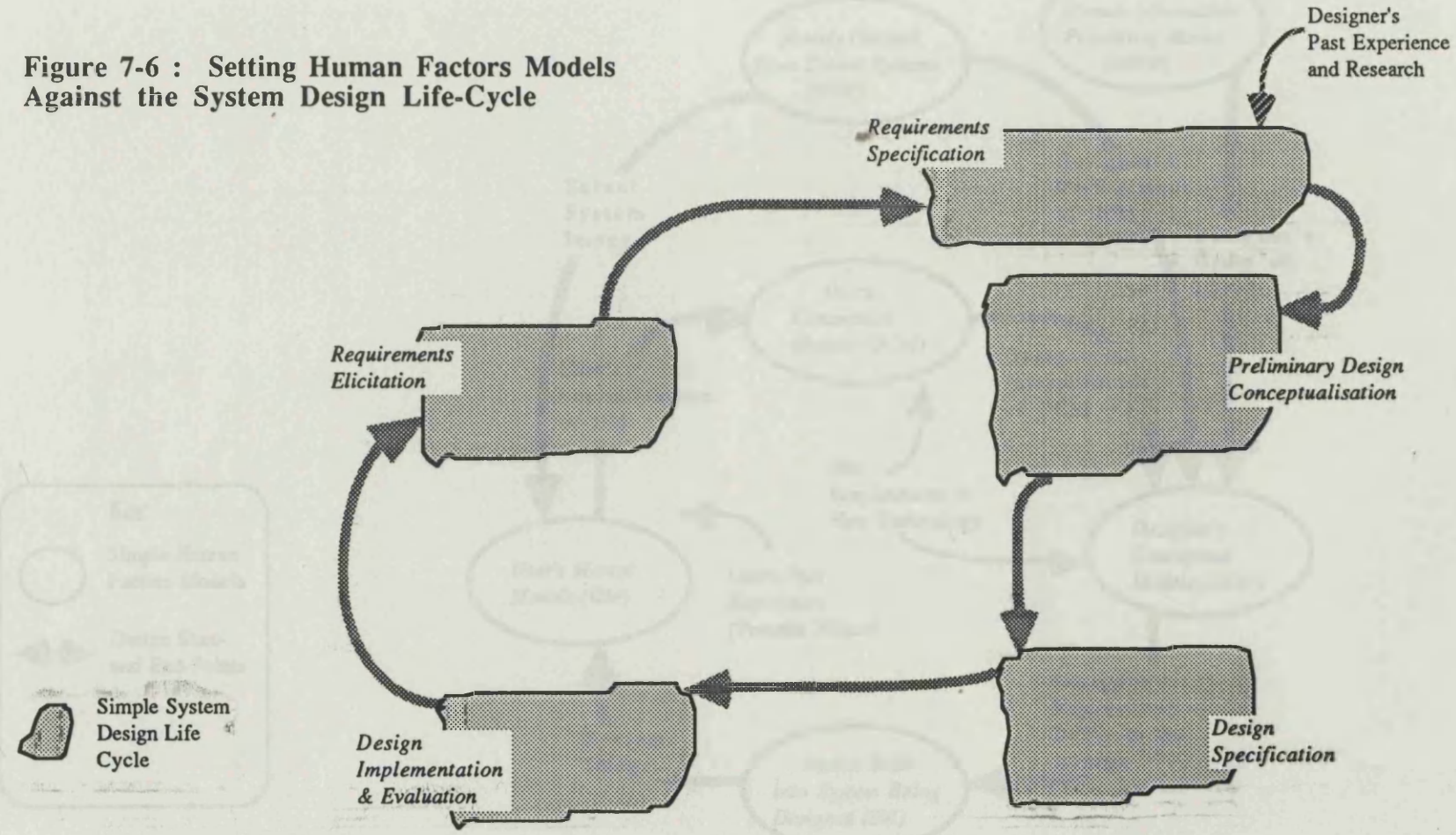
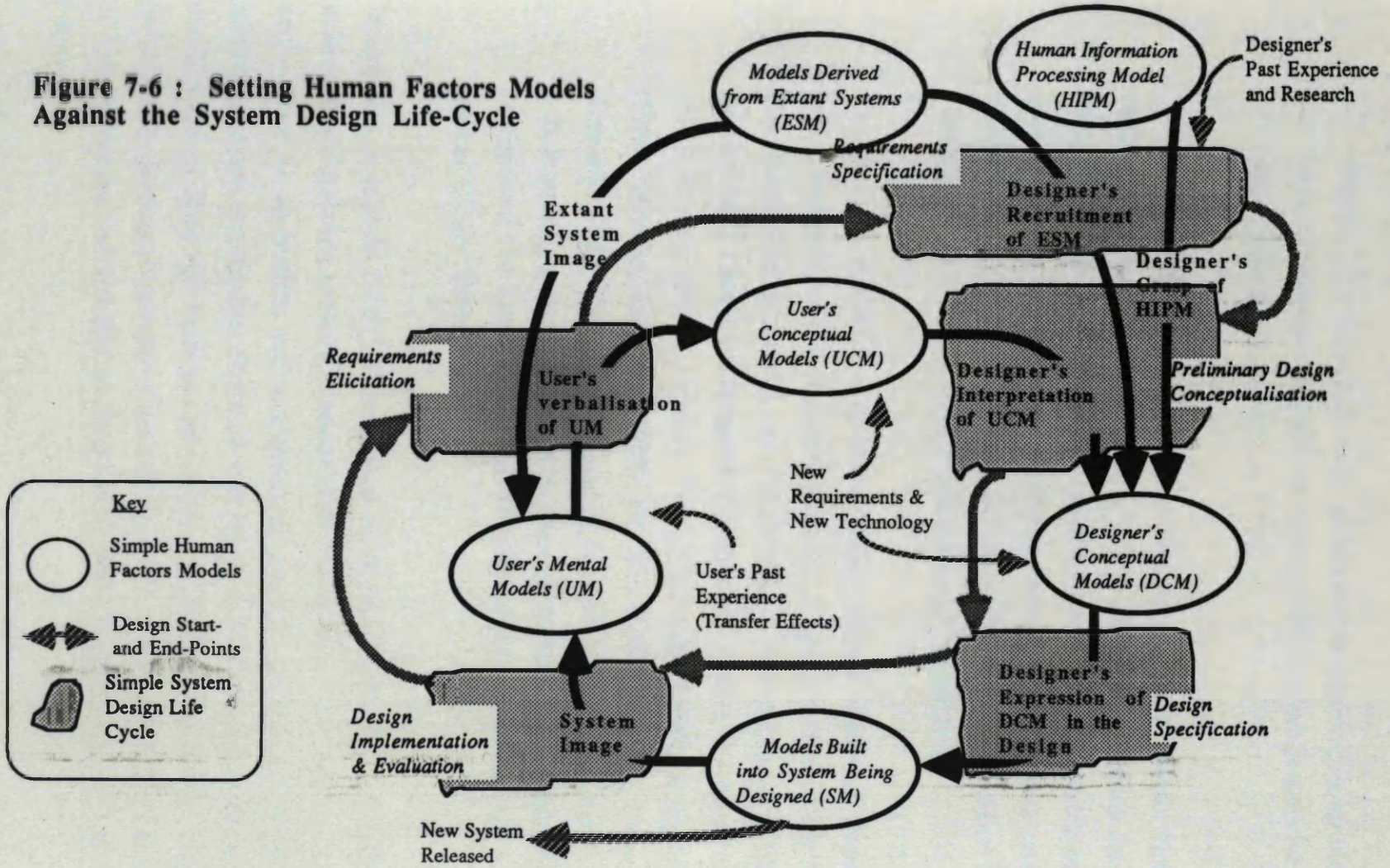


Figure 7-6 : Setting Human Factors Models Against the System Design Life-Cycle



details). The conception later supported the positing of an extant systems analysis technique and a user interface design process for the JSD*(HF) method (see Figure 7-7b).

In summary, the review revealed that the contributions of human factors models to system design is confined largely to late evaluation rather than to design specification. In addition, it indicated system and task models to be more promising recruits to the JSD*(HF) method, e.g. to support system design at the organisational level (e.g. socio-technical considerations) and interactive work system level (e.g. 'system image') respectively.

(f) case-study applications of human factors design techniques (human factors design 'practice'). The review included reports by Finkelstein and Finkelstein (1983), Smith et al (1982), Young (1988), Clowes (1986), Eason (1988), Farooq and Dominick (1988), Guest (1982), Hammond et al (1983), Moran (1981), Norman (1986), Newman (1988), Foley et al (1984), Frohlich and Luff (1989), etc. The techniques reported were compared with proposed conceptions of the JSD*(HF) method (see Table 7-7). Promising techniques were recruited and incorporated into successive conceptions. For a detailed account of these developments, the reader is referred to Lim (1989a, b).

In summary, a 'suitable' JSD*(HF) conception was derived on the basis of insights derived from literature reviews. Various case-studies were then introduced to demonstrate its application and to support an intensive phase of conception specification and test. Further literature reviews were conducted to support the rectification of observed inadequacies in each JSD*(HF) conception. Thus, progressively developed conceptions were derived with respect to the requirements of a structured method. Notable developments in this research phase include the following :

(1) conceptualisation of user requirements as comprising the specification of conditional statements pertaining to the task, device and environment (with

respect to the target user group);

(2) adoption of a user-task centered approach for system design. In this approach, design commences with a characterisation of the target user group and its current task performance on the extant system (denoted as 'System X' in Figure 7-7a). The information derived is compared with requirements that motivated the design of a new system. On this basis, appropriate extant task abstractions may then be recruited to advance the design via the synthesis of a conceptual task for the target system (denoted as 'System Y' in Figure 7-7a). These design concepts comprise an early version of the extant systems approach (see Lim, 1988d). Further decomposition of the conceptual task may then be undertaken to generate the user task model;[12]

(3) definition of an interaction task model to characterise inputs and outputs for a particular user interface design. The model is to be derived on the basis of conceptual and on-line task descriptions (corresponding to the composite

Table 7-7 : Comparison of Newman's (1988) Conception of User Interface Design 'Practice' and the JSD*(HF) Method

Newman's (1988) Conception of User Interface Design Practice		*Corresponding Stages of the JSD* (HF) Method*
(1) User Modelling Stage: specification of a mixture of user requirements, task knowledge and a design product similar to the domain of design discourse description advocated by the JSD*(HF) method.	⟷	(1) Extant Systems System Analysis and Statement of User Needs Stages : A user model is subsumed by the characteristics of task performance, i.e. task models are derived for a particular user group.
(2) The User Interface Style Design Stage addresses concerns of : (a) constraints imposed by style (b) suitability of particular styles (c) representation of style (d) extension and modification of styles	⟷	(2) Interface Model and Display Design Stages : -- extension and modification of chosen characteristics of the adopted user interface environment (if any).

[12] Note that the user task model was initially proposed as a description of *on-line* tasks only. On-line and off-line tasks remain undifferentiated until later in the research. Only then were the tasks described separately and re-named system and user task models respectively.

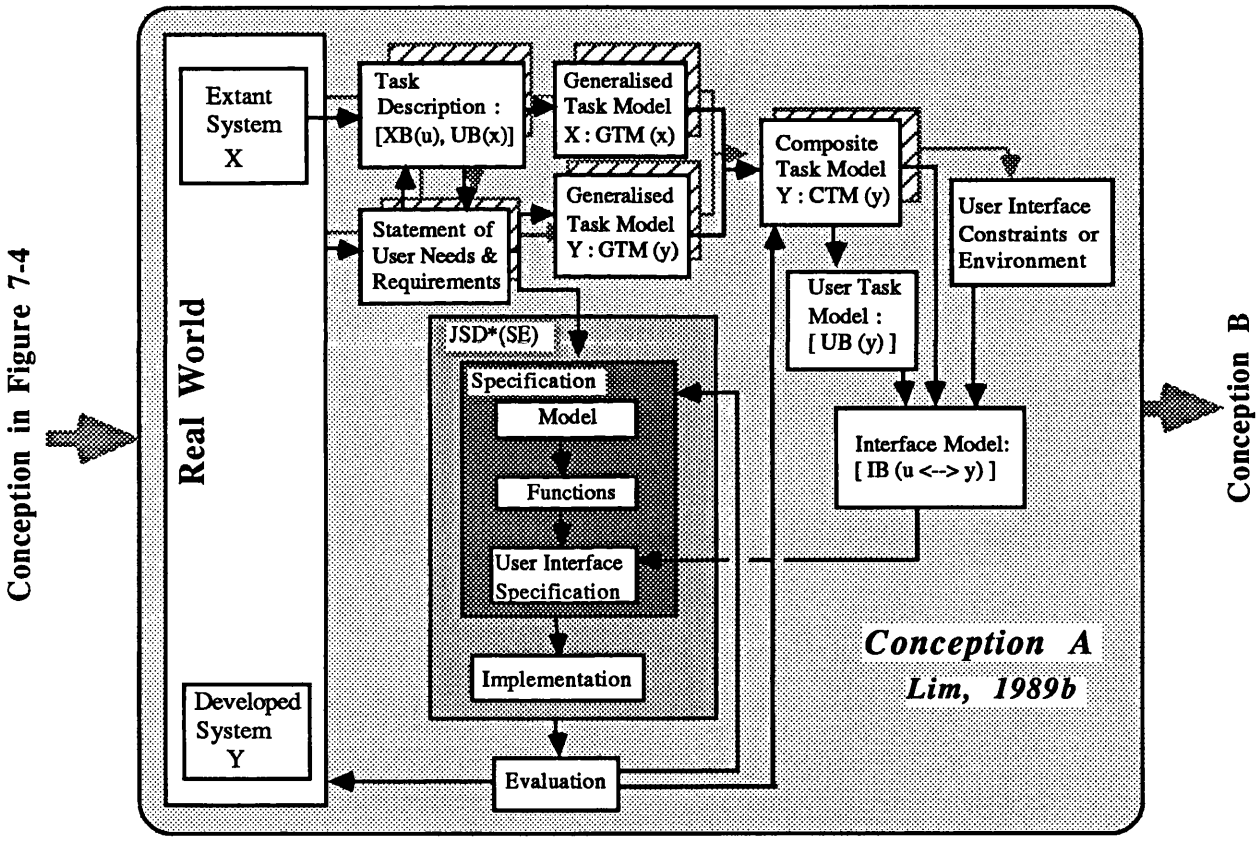
and system task models respectively), and the user interface environment or style (if any);

(4) propositions on how a suitable user interface environment or style could be selected using an extant systems analysis approach;

(5) conceptualisation of a set of 'basic elements' for describing a user interface design (see Figure 7-8);

(6) more explicit specification of the stage-wise design scope and processes

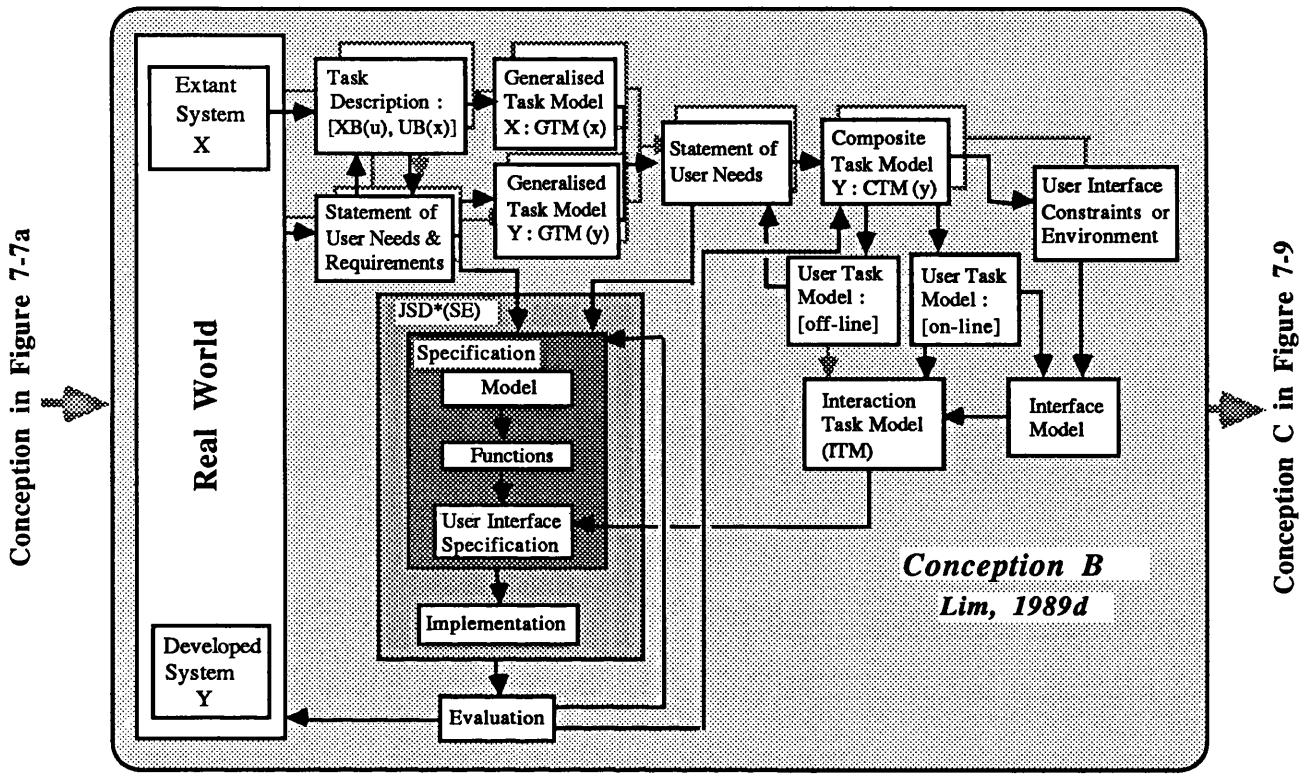
Figure 7-7a : Further JSD* Conceptions I



Following literature reviews, the JSD*(HF) conception in Figure 7-4 was detailed further. Four developments may be noted :

(a) an expansion of the task model to include generalised task model descriptions;
(b) further developments of the extant systems system analysis approach. Specifically, intermediate JSD*(HF) design products have been demarcated for both the extant system (System X) and target system (System Y);
(c) the introduction of separate descriptions of user and computer behaviours, corresponding to the derivation of a user task model and interface model descriptions respectively;
(d) the recognition of an earlier design cycle where an appropriate user interface environment (if any) may be assessed and adopted. This preceding design cycle is represented by boxes with oblique lines (see Figure).

Figure 7-7b : Further JSD* Conceptions II



Following further literature reviews and initial case-study tests, Conception A was developed further. Four developments may be noted :

(a) a stage-oriented representation of the conception (as opposed to the previous product-oriented representation);
(b) the separation of on-line and off-line user task components. User interface design is advanced primarily via the on-line task;
(c) the addition of an interaction level description of the user's task. This description corresponds to interface model descriptions of the computer behaviour;
(d) the distinction between explicit system requirements and implicit user needs. This distinction prompted the addition of a specific stage to summarise implicit user needs that have been elicited.

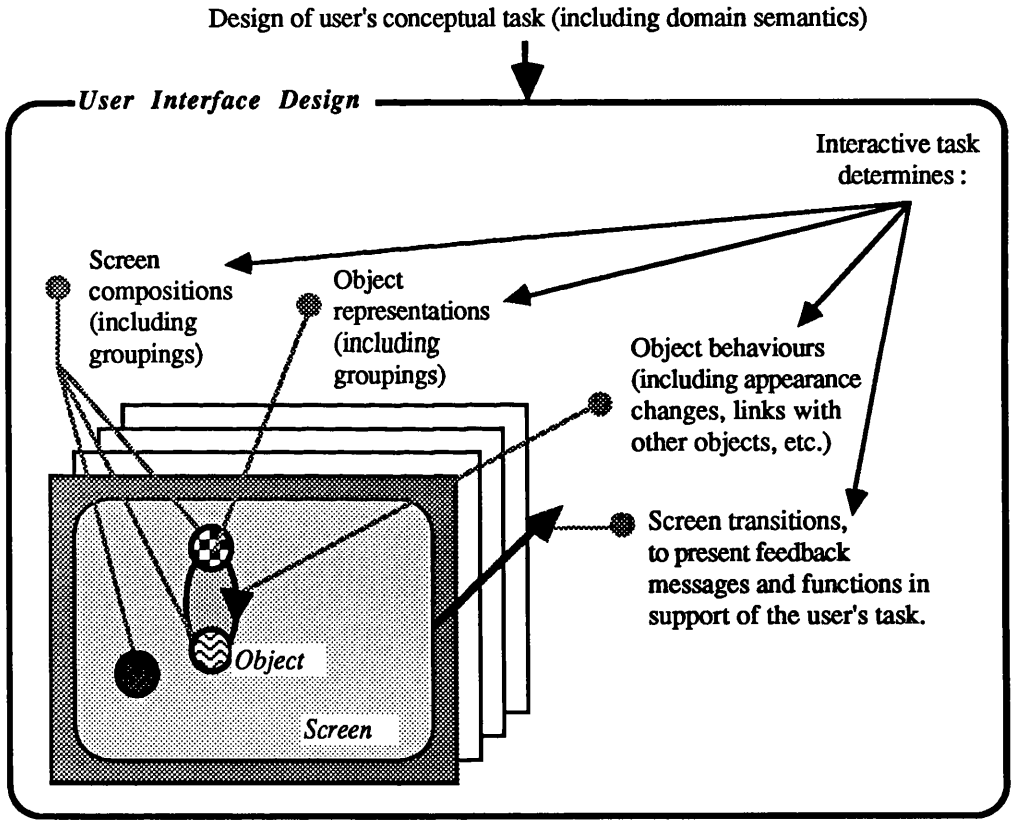
of the JSD*(HF) method. These characteristics are expressed respectively as a set of design products and procedures of the method. The proposed methodological developments were then tested in design case-studies involving three systems, namely the Digital Network Management System Simulator, Recreation Facility Booking System and Disk Operating System.™ Since these case-study systems represent various types of user interface designs (namely WIMP (which includes form-fill, menu and direct manipulation characteristics) and command-line user interfaces (Lim, 1989b)), a range of tests on the descriptive capability of JSD*(HF)

notations was thus afforded;

(7) compilation of a glossary of definitions and terms to support potential users of the JSD*(HF) and JSD* methods (see Part VII).

These developments culminated in the derivation of an 'acceptable' version of the JSD*(HF) method. Explicit integration with the JSD method was then attempted. These research concerns are described in the next sub-section.

Figure 7-8 : 'Basic Elements' for Describing a User Interface Design



7.1.3. A Review of Research Activities for Specifying Inter-Dependencies between Design Streams of the JSD* Method

The final phase of the research is concerned with specifying-and-testing various versions of the JSD* method. Thus, it is initiated by the integration of an

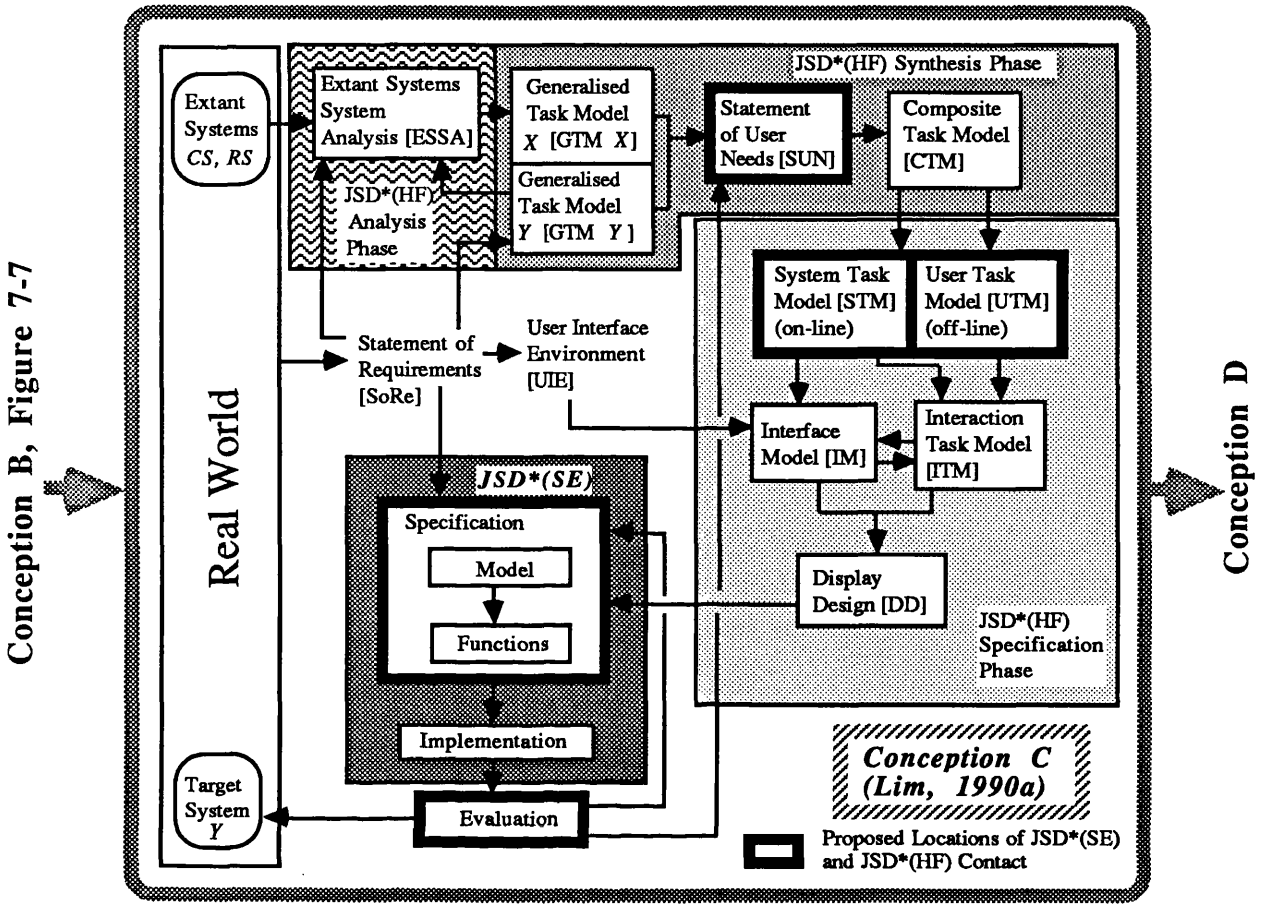
'acceptable' version of the JSD*(HF) method with the JSD method. Integration entails further examination of the JSD method so that design inter-dependencies may be located against appropriate stages of the JSD*(HF) method. Obligatory contact points between Software Engineering and Human Factors design streams were thus specified. The identification of these contact points was reasonably straightforward for the following reasons :

(a) the JSD*(HF) method was tailored specifically to an essentially unchanged JSD method. Thus, high level design intersections would have been addressed during the construction of the JSD*(HF) method;
(b) both JSD*(HF) and JSD methods were structured into explicitly defined design stages. Consequently, their design scope and processes (and hence information needs) could be intersected unambiguously.

The first attempt at integrating the methods was made following case-study tests using the Digital Network Management System Simulator. Design inter-dependencies between the two streams of the JSD* method were inferred by post-hoc comparisons of JSD and JSD*(HF) specifications of the case-study system (see Lim, 1989b). Thus, a version of the JSD* method that locates these design inter-dependencies explicitly was generated (see Figure 7-9a).

A breadth-wise test on a second case-study (namely the Recreation Facility Booking System) was then undertaken to verify the design inter-dependencies identified previously. To this end, both design streams of the JSD* method were undertaken in parallel and their design products were compared at these obligatory contact points. However, actual meetings between software engineers and human factors designers were considered unnecessary since the objective of the test was to verify *expected* and *actual* intersections in design information at inter-dependency points of the method. In other words, the forwards engineering test was concerned primarily with determining *what* information should be exchanged at *which* contact point, so that design convergence across the two JSD* streams may be ensured. Thus, a post-hoc comparison of design products from the two design

Figure 7-9a : Early Versions of the JSD* Method I



Intensive case-study tests were undertaken at this stage of the research. Findings from these tests supported further developments of the method, e.g. specification of later stages of the method. The following method developments should be noted :

(a) the Task Description Stage is now developed fully and re-named the Extant Systems System Analysis Stage;
(b) a clearer distinction between on-line and off-line components of the user's task was made. The on-line component was then re-named the 'system task model'. The 'user task model' is thus modified to comprise only the off-line component;
(c) procedures for the Interaction Task Model and Interface Model Stages were enhanced. In addition, the design procedures and products of the Display Design Stage were specified;
(d) design inter-dependencies between JSD*(HF) and JSD*(SE) streams were specified explicitly. Requisite information exchanges between the design streams were identified, e.g. intermediate design products and information to be shared;
(e) a clearer presentation of the method was considered, e.g. design activities of the JSD*(HF) method were organised into different design phases; the method scope was defined more tightly, e.g. a clearer location of 'secondary' design activities such as the selection of a suitable user interface environment, and wider concerns of requirements specification.

streams would be adequate (see Lim, 1989b).[13]

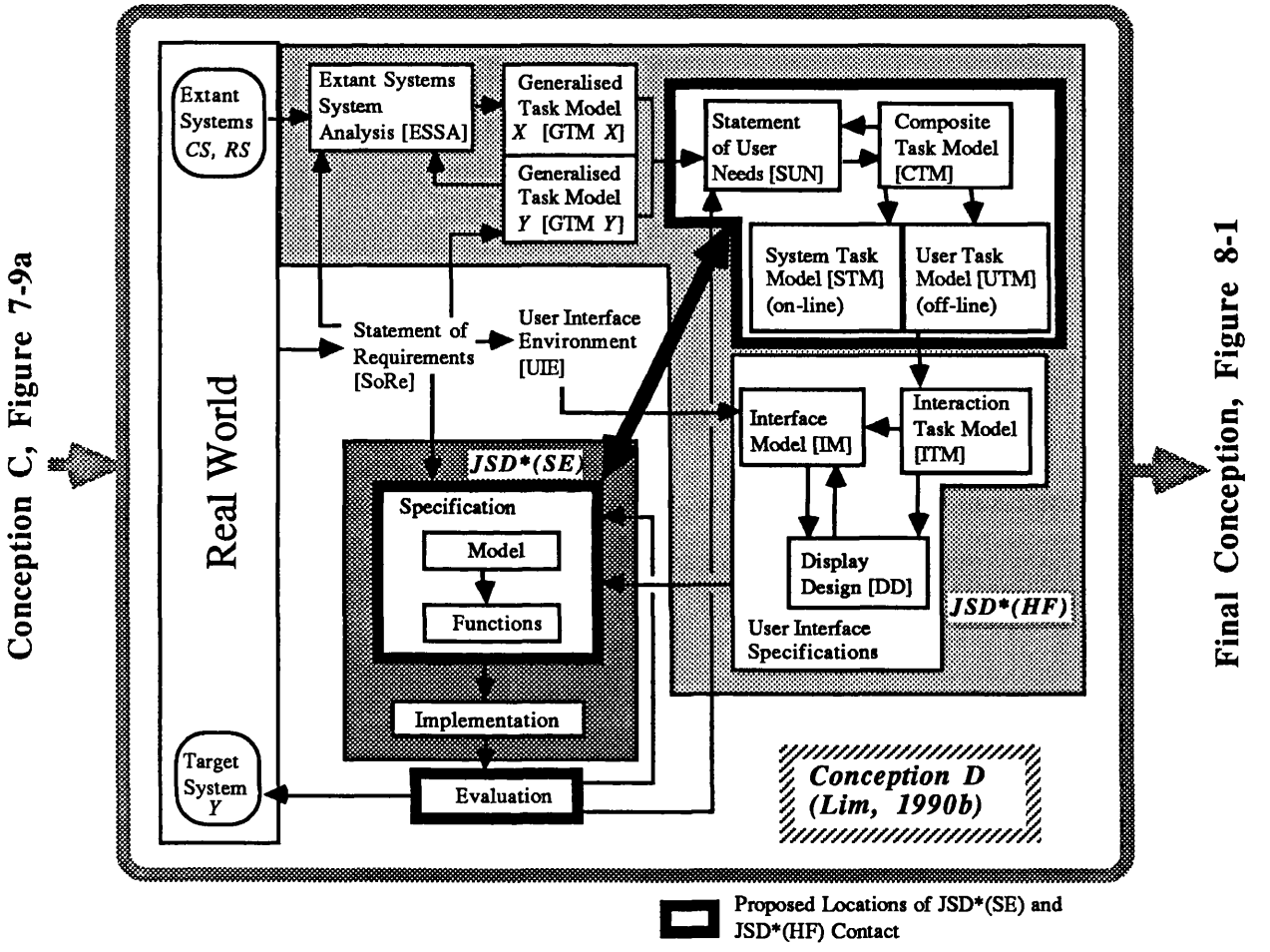
The pertinence of the proposed contact points was thus determined by assessing the design support afforded by sharing information at each of these points, e.g. how human factors design could benefit from accessing specifications of the input sub-system generated by software engineers and vice versa (e.g. how task descriptions generated by human factors (see Part IV) could define the context for specifying JSD functions. Minor modifications of the JSD*(HF) method instigated by these considerations were later implemented and tested in a second cycle using the Recreation Facility Booking System. For these tests, post-hoc comparisons of stage-wise JSD*(SE) and JSD*(HF) design products were repeated as before (see Lim, 1989d).

Succeeding versions of the JSD* method (see Figure 7-9a and b) were thus assessed repeatedly until an 'acceptable' version was derived. A larger case-study system, namely the Digital Network Management System, was then introduced to impose more demanding tests on the method. To this end, parallel test cycles were undertaken using the trouble-shooting and security modules of the case-study system (see Lim, 1990b, c, d, e; Silcock and Lim, 1990b; Silcock, 1990a, b). In contrast to preceding tests (see above), full application of the JSD* method was planned, i.e. both design streams of the method should be undertaken including the obligatory contact points. The objective of the tests was to demonstrate the JSD*(HF) method and to ascertain the appropriateness of contact points proposed for the JSD* method. Unfortunately, the tests could not be realised fully due to time constraints and excessive manpower demands from the sponsors of the research. Consequently, complete application of the method was discontinued following the first contact point, i.e. at the Composite Task Model Stage (see Part IV). Thus, case-study tests for assessing the appropriateness of proposed contacts points reverted subsequently to previous test scenarios, i.e. a

[13] The test scenario minimised resource commitments by the research sponsors with respect to their role as software engineers in the case-study. Potential delays to the research was thus minimised.

post-hoc comparison of JSD*(SE) and JSD*(HF) specifications. In addition, the case-study specifications generated by human factors designers and software engineers were not integrated explicitly for JSD implementation (although a user interface design prototype was constructed).[14] As a consequence, the final version

Figure 7-9b : Early Versions of the JSD* Method II



Intensive case-study testing was continued. Lessons learnt from these tests enhanced the method as follows :

(a) closer links were established between the Statement of User Needs and Composite Task Model Stages;
(b) the interaction task model was adopted as the basis for specifying interface model and display design descriptions;
(c) design inter-dependencies between JSD*(HF) and JSD*(SE) streams were made more explicit;
(d) human factors descriptions comprising the specification of a user interface design were defined explicitly.

[14] Despite these set-backs, the results of case-study tests on the JSD*(HF) method would remain valid (see also Chapter Twelve).

of the JSD* method (as opposed to the JSD*(HF) method -- see Footnote 14 and Figure 8-1) was not tested as comprehensively as would be desired (see Part V).[15] Nevertheless, overall indications from the case-study tests were considered positive.

The main product of the research may now be described. Since the existing JSD method (also referred to as the JSD*(SE) method to account for additional design inter-dependencies with the JSD*(HF) method) remains essentially unchanged, the account is focused on the JSD*(HF) method. In addition, an account of the JSD*(SE) method is unnecessary since its design inter-dependencies would be accommodated in a description of the JSD*(HF) method. However, an overview of the JSD* method is included to contextualise design contributions attributed to the stage-wise design scope, process and notation of the JSD*(HF) method. To illustrate the method, case-study examples are drawn from the security module of the Digital Network Management System.[16]

---

[15] This limitation is being addressed in a follow-up project commissioned by the research sponsors.

[16] This case-study module was selected because it is smaller than the trouble-shooting module and is thus more convenient for method illustration. It should be noted that original descriptions for the case-study have been revised and simplified to suit the purposes of the illustration.

PART IV :

A Structured Human Factors Method for the Jackson System Development Method

CONTENTS

Chapter Eight : An Overview of the JSD*(HF) Method.....188

8.1. JSD*(HF), JSD*(SE) and JSD* Methods.....188
8.2. General Characteristics of the JSD*(HF) Method.....190
8.3. Hierarchical Description of Work Systems and the JSD*(HF) Method.....194
8.4. Format for Presenting the JSD*(HF) Method.....195
8.5. Notes on the Choice and Scope of the Case-Study Illustration.....197

Chapter Nine : The Information Elicitation and Analysis Phase of the JSD*(HF) Method.....198

9.1. Extant Systems System Analysis (ESSA) Stage.....198
9.2. Generalised Task Model (GTM) Stage.....229

Chapter Ten : The Design Synthesis Phase of the JSD*(HF) Method.....239

10.1. Statement of User Needs (SUN) Stage.....239
10.2. Composite Task Model (CTM) Stage.....247
10.3. System and User Task Model (SUTaM) Stage.....262

Chapter Eleven : The Design Specification Phase of the JSD*(HF) Method.....273

11.1. Interaction Task Model (ITM) Stage.....273
11.2. Interface Model (IM) and Display Design (DD) Stages.....281

Chapter Eight : An Overview of the JSD*(HF) Method

"In the land of the blind, even the one-eyed man is king."
bottom-line argument for the method? John Long, 1990.

"Good order is the foundation of all good things."
Edmund Burke, 1790.

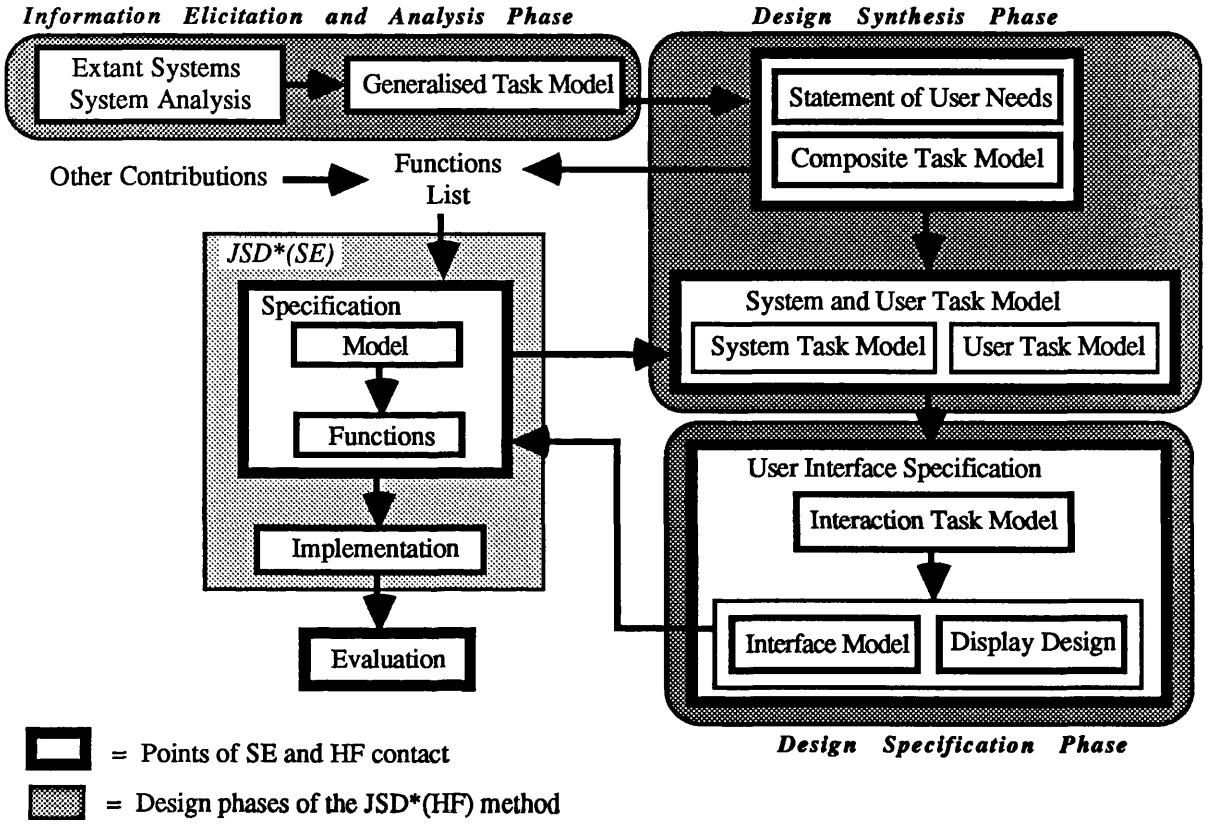
The objective of the present overview is to establish a conceptual foundation for a detailed account of the JSD*(HF) method. The latter account comprises a stage-wise description of the method and its relationship with the JSD*(SE) method (see later chapters).

8.1 JSD*(HF), JSD*(SE) and JSD* Methods

JSD*(HF) is one of two component structured methods that constitute the JSD* method (see Figure 8-1).[1] It is the human factors component of JSD* developed specifically for integration with an essentially unchanged JSD method. Henceforth, the latter method will be referred to as the JSD*(SE) method since additional inter-dependencies with the JSD*(HF) method have been specified to support collaborative design. These inter-dependencies constitute obligatory contact points at which design information is shared and agreed between designers working in the parallel streams of the JSD* method (inter-dependencies are indicated in Figure 8-1 by boxes outlined in bold). Once agreed, the information becomes binding since it constitutes the basis for later design extensions. Thus, any violation of the agreed design basis should be communicated to designers working in the other stream of the JSD* method. In this way, a common design focus and scope may be ensured for SE and HF design to proceed in parallel.

[1] Figure 8-1 shows that the JSD*(HF) method comprises the focus of the research. For this reason, its design stages are shown in greater detail relative to the JSD*(SE) method. Actual differences in complexity between the methods are not represented in the Figure.

Figure 8-1 : Locating the JSD*(HF) Method within the JSD* Method



In addition to design inter-dependency points, other meeting points may be defined in a given application of the JSD* method. Such meetings are usually dictated by situation- or organisation-specific factors, e.g. meetings instigated by *ad hoc* factors such as the close proximity of design team members; information quality at project inception; particular local requirements; etc. Thus, to ensure general applicability of the method only obligatory design inter-dependencies between the streams are emphasised. In other words, contact points which are not fundamental for advancing the design are considered discretionary (i.e. not a requirements of the method) and are accommodated only when necessary. The objective is to ensure that JSD* is not over-specified so ensuring its general applicability and flexibility (although it should be adequately specified to meet the requirements of a structured method). For instance, it should be sufficiently flexible to accommodate adaptations in its application. An example of flexibility in the method may be illustrated by the Extant Systems System Analysis Stage where the depth of extant systems analysis

is specified as a 'bottom-line' requirement, i.e. the derivation of a current task description is obligatory while other design products are derived only if the designer considers them necessary. These methodological concerns are discussed at length in Chapters Nine and Twelve.

8.2 General Characteristics of the JSD*(HF) Method

Generally, the focus of the JSD*(HF) method is on complementing the design specification stages of the JSD*(SE) method. Such a focus is appropriate because :

(1) literature surveys indicate that current HF contributions to system development are well established at later stages of the design cycle, e.g. HF evaluation during design implementation. In contrast, HF contributions to design specification are inadequate and implicit. Since the recruitment of HF contributions is traditionally late, the discovery of such design errors is also delayed. As a result, the required modifications are costly and difficult to implement (see Chapter One). Thus, greater emphasis should be placed on ensuring HF contributions to design specification;
(2) JSD*(SE) implementation activities comprise mechanistic transformations which do not alter design specifications defined at earlier stages of the JSD* method. In other words, the external behaviour of JSD*(SE) specifications is maintained if the transformations comply with implementation rules of the method (Zave, 1984). Thus, HF contributions at JSD*(SE) implementation stages would be minimal if appropriate HF inputs have been incorporated during design specification. In particular, HF contributions at design implementation would be confined to the following :

(i) designing additional feedback cues for users if longer than expected transient response times result from a particular JSD*(SE) implementation (e.g. a feedback cue such as the Macintosh wrist-watch icon may be provided). This design scenario may arise if hardware specifications (e.g. processing capabilities) can not be met subsequently due to budgetary constraints (e.g. the number of

processors available is lower than expected), or unforeseen hardware limitations;

(ii) validating the fidelity of JSD*(SE) implementation. The proposed transformation of JSD*(SE) specifications may be evaluated by conducting user tests on either a design prototype or the final artefact;

(iii) ensuring appropriate design modifications (and design iteration as necessary) following evaluations in (ii) above.

For these reasons, a participative followed by consultative HF role is envisaged at JSD* design specification and implementation stages respectively. In respect of the latter, existing evaluation methods and practices may be recruited to support the method. An overview of the JSD*(HF) method follows.

The JSD*(HF) method is structured into three *phases*, each of which comprises a number of design *stages* (see Figure 8-1). The scope of the design *phases* is as follows :

(i) the *Information Elicitation and Analysis Phase* is concerned with user requirements capture and task analysis. Its design stages comprise the Extant Systems System Analysis and Generalised Task Model Stages;

(ii) the *Design Synthesis Phase* addresses the derivation of a conceptual design of the target system. Its design stages comprise the Statement of User Needs, Composite Task Model and System and User Task Model Stages;

(iii) the *Design Specification Phase* is focused on functional and user-interface design. The remaining three stages of the JSD*(HF) method, i.e. Interaction Task Model, Interface Model and Display Design Stages, belong to this phase.

A detailed account of the JSD*(HF) method is presented in later chapters. It suffices to say at present that salient characteristics of the method include the following :

(1) its stages comprise coherent groupings of design processes which

transform inputs to desired outputs. In most cases, the output or outputs of a design stage constitute input or inputs to a succeeding stage. Wider relationships may also apply for a number of stages, e.g. design output(s) of one stage may feed into several succeeding design stages. Such instances are indicated explicitly when the stages are described in detail later;
(2) its stages are defined explicitly with respect to their scope, process and notation. Thus, each design stage is characterised by a set of design products, procedures and documentation schemes;
(3) its stage-wise design products are explicitly defined. Thus, prototyping is encouraged to accommodate the incompleteness of current HF knowledge (see also (4) and (5) below). To this end, prototypes may be constructed at each stage of the method to exemplify proposed designs. Consequently, early and continuous evaluation is encouraged throughout the system design cycle. However, as with any structured method, the JSD*(HF) method may be incompatible with the design approach entailed by rapid prototyping. In particular, a structured method involves a phase of design analysis and documentation prior to the specification of a 'first-best-guess' solution (which may then be prototyped). Such a design phase is excluded in a rapid prototyping approach;
(4) its methodological structure upholds accepted design principles such as :

(i) the delaying of design commitments (Thimbleby, 1990), e.g. by ensuring that detailed design is preceded by an adequate conceptual design;
(ii) the conduct of early design evaluation either analytically by the designer or empirically using a prototype;
(iii) the conduct of iterative design;
(iv) the conduct of incremental or modular design development. This design scenario may be pursued in both streams of the JSD* method following the specification of a conceptual design;

(5) its well defined stage-wise scope, process and notation provide a basis for configuring and recruiting alternative means of HF input, e.g. guidelines, computer-based tools and prototyping. For instance, its well

defined stage-wise scope facilitates the identification of appropriate design guidelines for recruitment throughout the design cycle. Similarly, its well defined stage-wise scope, process and notation provide a basis for developing computer-based tools to support design specification. Finally, prototyping is encouraged at each stage of the method since its design products are all defined explicitly (see (3) above);

(6) its notation is essentially an adapted version of the structured diagram notation of the JSD*(SE) method (see Annex A and Figure 6-1). The notation was recruited following a series of assessments which indicated its suitability for describing HF design products (Walsh, 1987a; Lim, 1988e; Carver, 1988). The structured diagrams are supported by tables which provide a detailed textual description;

(7) its methodological configuration is tailored specifically to the JSD*(SE) method since the latter is left essentially unchanged. In particular, design inter-dependencies between the JSD*(SE) and JSD*(HF) methods are defined to support collaborative design. Assimilation of the JSD*(HF) method by software engineers may also be facilitated since familiar reference points within the JSD*(SE) method are highlighted by the design inter-dependencies. Thus, a positive transfer of knowledge is supported; [2]

(8) its procedures are targeted at a HF designer with a basic understanding of the JSD*(SE) method. Experience in the latter method over and above this general requirement may enable a more effective realisation of the JSD* method. However, it should be noted that a particular design team composition is not implied by the JSD* method. The only implication of the method is that HF, SE and domain expertise should be represented. Thus, later references to design teams working in respective streams of the JSD* method, are for explanatory purposes only. While design teams may be the case on most occasions, the undertaking of both method streams by a single designer should not be precluded;

(9) its approach to design specification is best characterised as a user-task

[2] Maintaining an unchanged JSD*(SE) stream was a requirement set down by the sponsors of this research (see acknowledgements and Chapter Six).

oriented approach.[3] During design, a user model is realised in terms of particular task execution and performance characteristics;

(10) its focus is primarily on design specification rather than on design implementation and evaluation. The bias is intended to redress the imbalance arising from the traditionally late recruitment of human factors contributions to system development.

The above account completes an overview of the JSD*(HF) method. A detailed description of the method is presented in Chapters Nine to Eleven.

8.3 Hierarchical Description of Work Systems and the JSD*(HF) Method

Large human-computer systems usually involve a co-ordinated network of interactive sub-systems, e.g. tactical planning and control modules of defence systems on board an aircraft carrier. In such cases, the method should be applied in two steps as follows :

(i) conceptual design at the organisation level is first defined in the first two phases of the method, i.e. the Information Elicitation and Analysis, and Design Synthesis Phases. Following a conceptual definition of the system purpose, sub-systems may be identified. Formal socio-technical interactions[4] among the sub-systems (e.g. work relationships and information exchanges) are then described using a mixture of data flow and network diagrams (used in SSADM and SASD, and JSD respectively). The descriptions derived at these phases of the JSD*(HF) method complements similar information derived by JSD*(SE) designers;

(ii) lower level requirements, conceptual and functional design of each sub-system may then be specified. To this end, the first two phases of the

---

[3] See Fleishman and Quaintance (1984) for alternative perspectives on human task performance.

[4] Although informal work relationships are difficult to uncover, they should be addressed during design when appropriate.

method are repeated to define each sub-system more specifically. Design specifications for the sub-system is then derived by a complete application of the method. The process is repeated until design specifications for all sub-systems are derived.

Since JSD*(HF) activities at the organisation level are repeated at the sub-system level, the method may be exemplified completely by a description of the latter (see later chapters).

8.4 Format for Presenting the JSD*(HF) Method

In accordance with the definition of a structured method, the JSD*(HF) method is presented in terms of its stage-wise design scope, process and notation. In particular :

(i) the *scope* of each of its stages is described in terms of the design products to be derived. To illustrate the products, examples are selected from a case-study undertaken during method development, namely a Digital Data Network Management System;
(ii) its design *process* is characterised at two levels of description, namely at inter-stage and intra-stage levels. The former comprises a stage-to-stage description of the method while the latter entails the definition of sub-processes and procedures for each of its design stages (targeted at a HF designer with a basic understanding of JSD). Intra-stage level processes and procedures are described using a block diagram (see Figure 8-2) and text respectively. Rules of thumb (i.e. semi-formal notes) to support method application are also included where appropriate;
(iii) the *notation* (including documentation formats) for describing human factors design products is defined for each of its stages. Illustrations of such description schemes are provided using examples selected from the case-study cited in (i) above.

To facilitate method assimilation, the above presentation format is supported by the

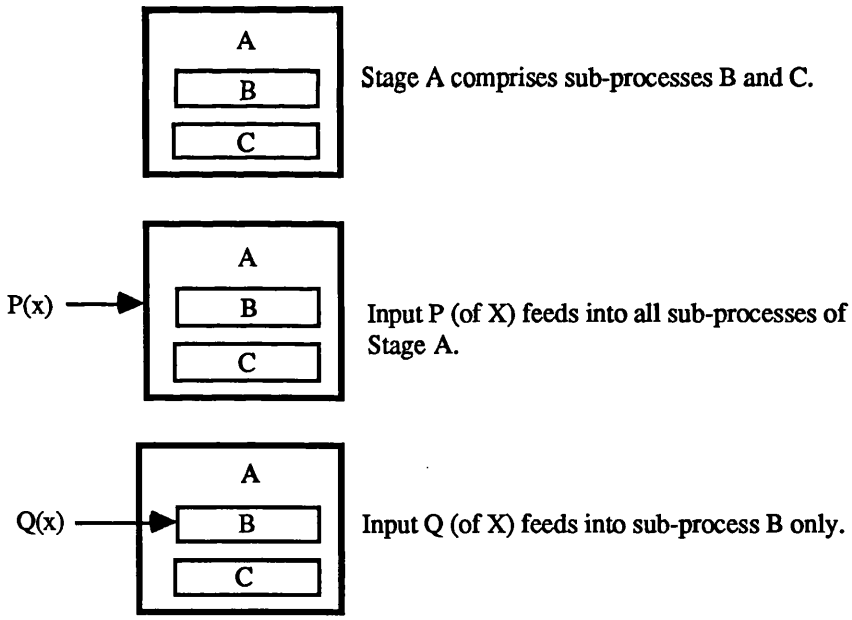
following :

(a) an overview of each stage is provided for readers who are generally interested in the method;

(b) a detailed account of the design stage follows the overview in (a) above. Design products, procedures and notations of the stage are described and illustrated using case-study examples. Thus, a deeper understanding may be derived by readers who may be interested in applying the method.

In summary, the presentation format permits selective reading of Chapters Nine to Eleven corresponding to the level of method exposure desired by the reader.

Figure 8-2 : Block Diagram Summary of Each Design Stage of the JSD*(HF) Method



8.5. Notes on the Choice and Scope of the Case-Study Illustration

Many case-studies were undertaken in the course of the research. However, the final case-study (concerning the design of a network management system) is most suitable for illustrating the JSD*(HF) method.[5] Since the scope of the system is rather large (relative to resources available to the project), only the trouble-shooting and security modules were addressed during method development. Of the two modules, a sub-set of the design descriptions for the security module is used to illustrate the method. This sub-set is selected to meet the following requirements :

> (a) it should adequately illustrate the stage-wise design scope and notation of the method. This requirement includes a comprehensive illustration of the the products and documentation schemes of each design stage;
> (b) it should adequately exemplify design transformations between stages of the method. This requirement implies that case-study examples should illustrate a coherent and traceable thread of design advancements across the stages of the method.

In other words, the objective of the case-study is to expose the procedural human factors knowledge embedded within the JSD*(HF) method. However, the recruitment and application of declarative human factors knowledge are excluded from the scope of the case-study illustration since they fall outside the remit of the present research. Nevertheless, such instances are highlighted in the illustrations when appropriate.

Against this background, a detailed account of the JSD*(HF) method is presently described.

---

[5] The assertion stems from the strategies for case-study tests on the method (see Chapter Six). In particular, the strategies require that the final case-study be a non-trivial system so that a more complete test of the method is afforded. Thus, the final case-study is most suitable for illustrating the method.

Chapter Nine : The Information Elicitation and Analysis Phase of the JSD*(HF) Method

"Really we create nothing. We merely plagiarise nature."
Jean Baitaillon.

"Engineering attempts to fully constrain its outputs.....Engineering investigates successful designs and adopts those 'means' that it finds generalisable."
Jim Carter.

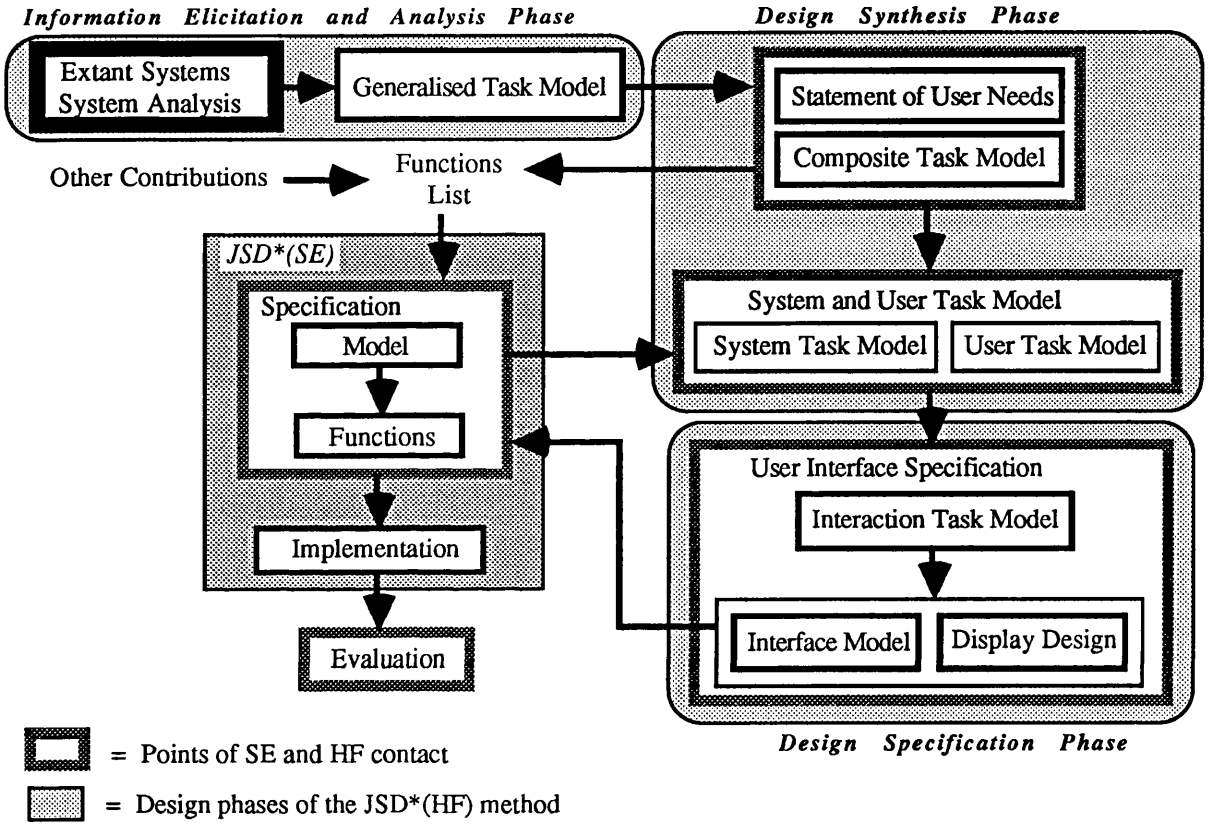
This chapter presents a stage-wise account of the Information Elicitation and Analysis Phase of the JSD*(HF) method. The two stages of this phase, namely the Extant Systems System Analysis and Generalised Task Model Stages, are described in the order by which design is advanced (see Figure on the following page). The stages are concerned with the generation and analysis of background information that would later support the derivation of a design solution. Intermediate processes and products for each stage of the method are described in the format proposed in Chapter 8. Specifically, each stage of the method is expanded graphically as a block diagram containing one or more sub-processes. The stage-wise processes transform inputs into a number of intermediate design products. Case-study examples of these design products and processes are provided where appropriate. In addition, design relationships between the JSD*(HF) and JSD*(SE) method are highlighted.

9.1. Extant Systems System Analysis (ESSA) Stage

Summary

The main objective of the ESSA Stage is to generate background design information that will assist target system design. In analysing extant systems, the HF designer may be interested in characterising current user needs and problems; existing task allocation between the human and device; existing user interface design features and rationale; etc. The information is described by a number of ESSA Stage products, each providing a different perspective on the design, e.g. user-task design, user

interface design, etc. (see later and Annex D). These products constitute the data base that supports target system design at later stages of the method (see Figure below -- the ESSA Stage is highlighted in black).



Two aspects of extant systems analysis should be noted. Firstly, *extant* systems include both the *current* system (i.e. the system currently in use in the client organization) and *related* systems (i.e. similar systems in use in other sections of the client organisation or elsewhere). The objective of including related systems in the initial analysis (as opposed to studying only the current system) is to avoid 'tunnel vision' at an early stage of system design. It is expected that analyses of extant designs would provide valuable insights about appropriate and inappropriate design features with respect to the target system, i.e. extant designs are assessed on their potential for recruitment to the target system. In practice, extensive analysis of the current system and a number of extant systems may be involved (see below). Thus, a broader perspective is derived to support wider consideration of alternative

designs. In addition, information derived from extant systems analysis may indicate possible transfer of learning by current system users (both positive and negative transfer). These concerns are addressed later at the Generalised Task Model Stage.

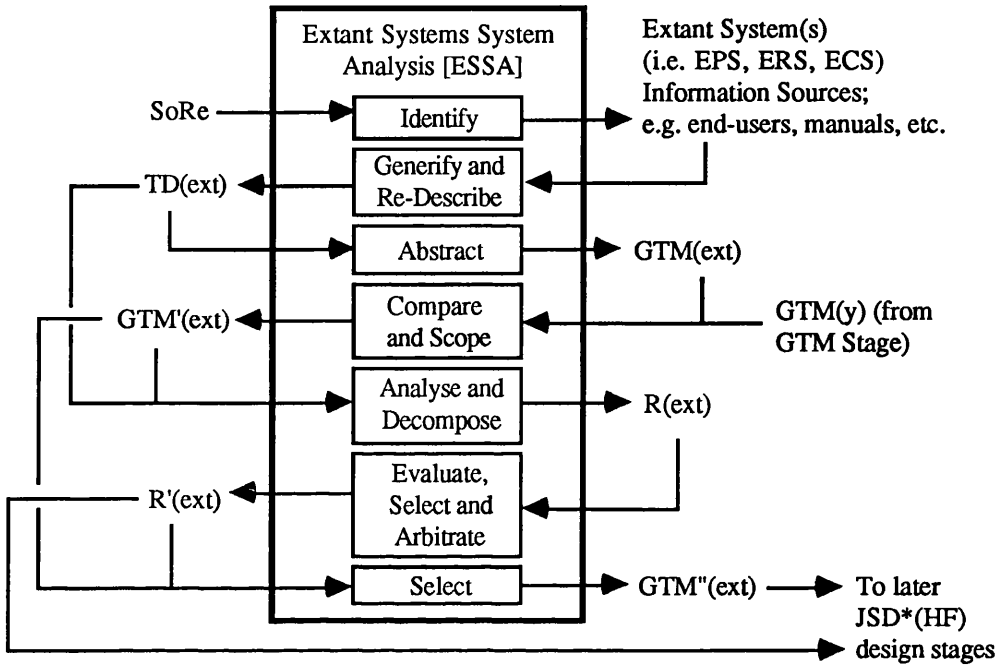
Secondly, it should be emphasised that the detail to which extant systems are analysed depends on the circumstances of the design project, e.g. the resources available; the designer's familiarity with the system domain; etc. Indeed, the study of a related system need not be undertaken physically if the designer is able to draw on past experience of related systems. In other words, the extent of such analyses should be decided by the designer in each instance. Where limited extant systems analysis is undertaken, the designer is required to note the information source and rationale of all products leading to a target system design.

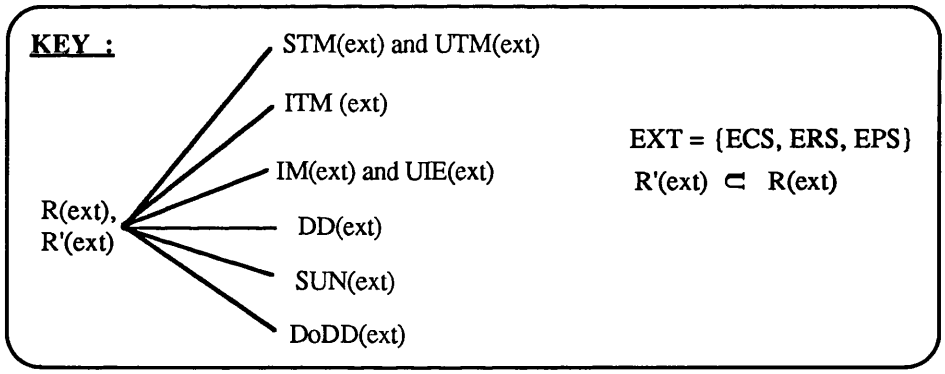
A brief account of the design processes and products of this stage follows (see Figure 9-1. The reader is referred to Chapter 8 for an explanation of the representation scheme). Italics will be used to highlight the design processes shown in the Figure.

An initial statement of requirements that describes key target system characteristics is collated from the client's brief, contractual documents, etc. On the basis of such a statement, relevant extant systems may be *identified* at the ESSA Stage. Thus, appropriate current, related, and partially related systems are selected for analysis. Using 'off-the-shelf' techniques such as interviews, unobtrusive observations, etc., pertinent information on these systems may then be elicited from various sources, e.g. end-user groups, existing job descriptions, user manuals, etc.

Structured diagram descriptions of two primary products are derived for each extant system analysed. First, a Task Description for the extant system (TD(ext)) is derived by decomposing tasks into sub-tasks. In most cases, a single task description would have to be collated from information elicited from diverse sources, e.g. protocols with different task performers, etc. To achieve this objective, a basic set of generic descriptors or 'building blocks' (e.g. objects and actions that are common across the information sources) needs to be defined. Thus, *generification* procedures suggested by Johnson and Johnson (1987); and

Figure 9-1 : Block Diagram Summary of the Extant Systems System Analysis (ESSA) Stage





DD = Display Design
DoDD = Domain of Design Discourse
ECS = Extant Current System
EPS = Extant Partial System
ERS = Extant Related System
(ext) = extant system (EXT) descriptions
EXT = Extant Systems
GTM = Generalised Task Model
IM = Interface Model
ITM = Interaction Task Model
R = JSD(HF) design products*
SoRe = Statement of Requirements
STM = System Task Model
SUN = Statement of User Needs
TD = Task Description
UIE = User Interface Environment
UTM = User Task Model

Johnson, Johnson and Russell (1988), have been recruited to support the JSD*(HF) method. Similarly, to support later design analysis and synthesis an appropriate level of task description needs be derived. Thus, the method includes conditions which state that task decomposition should be terminated only when the description derived :

(a) is generally understood among designers and end-users. In particular, tasks should be decomposed to a level that facilitates unambiguous identification with pre-specified system goals;
(b) is commensurate with the criticality, frequency and centrality of the task (see Johnson and Johnson, 1987);
(c) supports unambiguous identification of performance shaping factors (PSFs);
(d) supports the identification of training requirements and criteria, e.g. the attainment of satisfactory Probability of failure x Cost (or P x C) values (Duncan, 1974).

Second, a Generalised Task Model for the Extant System (GTM(ext)) is *abstracted* from the Task Description (TD(ext)). The objective is to remove device dependent details to facilitate later comparisons between extant and target system designs. Although the level of abstraction should be sufficiently high to reveal the logic underlying the system task, the resulting description need not be homogenous. For instance, the level of abstraction may be deliberately lower for particular parts of the description to preserve information on design features of interest in later stages. Thus, the generalised task model supports HF analysis of extant system designs vis-a-vis requirements of the target system.

Apart from the above products, other ESSA Stage products may be derived depending on individual project circumstances. For instance, the full complement (comprising products of all subsequent stages of the method -- the set is represented generally as R(ext) in Figure 9-1) may be derived in cases where the extant and target systems are highly similar (as in variant design). However, it is unlikely that all extant system information will be relevant to target system design although extant systems are selected on the basis of common domain or task characteristics. Thus, an appropriate *scope* of analysis is identified by *comparing* GTM(ext) with

the Generalised Task Model of the Target System (i.e. GTM(y) -- see later). A relevant sub-set of GTM(ext) (denoted as GTM'(ext) in Figure 9-1) is thus identified. Both TD(ext) and GTM'(ext) support the generation of further ESSA Stage products as appropriate. For instance, TD(ext) and GTM'(ext) may be *analysed* further to identify relevant R(ext) descriptions. Lower level *decompositions* such as STM(ext) and ITM(ext) may then be collated.

Generally, the R(ext) descriptions derived are *evaluated* to identify potential candidates for recruitment during target system design. The candidate set is denoted as R'(ext) descriptions in Figure 9-1. Since R'(ext) descriptions relate to a sub-set of GTM'(ext), pertinent parts of the latter are collated to derive a GTM"(ext) description for each of the extant systems analysed. The set of GTM"(ext) descriptions are then carried forward to the next stage (the Generalised Task Model Stage) where compatible aspects may be synthesised on the basis of the statement of requirements and GTM(y). Thus, an overall GTM(x) description is derived.[6] Similarly, other R'(ext) descriptions feed later stages of the method where corresponding JSD*(HF) descriptions of the target system are derived, e.g. SUN(ext) descriptions support the derivation of SUN(y) descriptions at the Statement of User Needs Stage, etc. In other words, ESSA Stage information is processed into products whose scope and format are similar to corresponding products derived at later JSD*(HF) stages (see Figure 9-1). For instance, the information is processed into descriptions comprising JSD* structured diagrams, semantic nets, pictorial diagrams and tables. Uptake of ESSA Stage products is thus facilitated.

Since ESSA Stage analysis may appear to address 'variant' design only,[7] it is pertinent to emphasise that the method is not limited to such designs. On the contrary, together with the Generalised and Composite Task Model Stages, 'novel' design is also supported (see later).

---

[6] JSD*(HF) descriptions with an '(x)' suffix denote products synthesised from parts of extant system descriptions (denoted by an '(ext)' suffix). Similarly, design descriptions with a '(y)' suffix denote JSD*(HF) products associated with the target system.

[7] Note that most system designs currently involve variant design (see Rouse and Boff, 1987).

A more detailed account of the ESSA Stage, appropriate for potential users of the method, follows.

Detailed Account

ESSA Stage activities may be grouped into three categories, namely :

(a) Identification of extant systems for analysis
(b) Elicitation of task information
(c) Derivation of ESSA Stage products

A review of the design activities and their procedures follows. Case-study examples of ESSA Stage products are also included as appropriate.

(a) Identification of Extant Systems for Analysis

The first step in the method is to characterise the purpose of the target system (this step is undertaken together with JSD*(SE) analysts). Thus, information pertaining to the subject matter and requirements of the system is extracted from the client's brief, e.g. target system tasks, hardware requirements, desirable current system characteristics and perceived future needs. Other information sources such as feasibility reports, informal and contractual documents, transcripts of protocols with stake-holders, etc., may also be consulted.

The information elicited is summarised to generate an initial statement of requirements. The statement should be sufficiently detailed to define the scope of target system design. In particular, it should identify key characteristics of the target system such as the domain of application (e.g. Network Management), technological constraints (e.g. a command-line user interface), end-user characteristics (e.g. novices), etc.

Target system characteristics thus identified are grouped into one or more sets of

'concrete' and 'abstract' characteristics which are of particular interest to the HF analyst. 'Concrete' sets map onto extant systems that operate in the same domain as the target system, i.e. 'variants' of the target system. 'Abstract' sets, on the other hand, map onto systems operating in different or partially similar domains. These systems may be compared conceptually with the target system to support 'novel' design (as opposed to 'variant' design). However, the analysis of such systems would not generate the same wealth of information since its relation to the target system is more distant than 'concretely' related systems.

On the basis of these sets, suitable extant systems may be identified for analysis. In particular, three categories of extant systems may be identified, namely :

(1) the 'extant *current* system' in use in the client organisation, i.e. the system to be replaced by the target system. Aside from domain similarity, device characteristics of the current system may or may not be related to the target system. For instance, a weak relationship would be expected in computerising manual systems. It should be noted that the current system is especially important from a HF viewpoint as transfer effects (both positive and negative) attributed to current expectations and experiences of installed users would have to be addressed during target system design;
(2) 'extant *related* systems' whose domain of application is *similar* to the target system. This category includes systems in use in other sections of the same organisation or elsewhere;
(3) 'extant *partial* systems' which share sub-tasks similar to those intended for the target system, but whose domains of application are largely *different* or *unrelated*.

The relationship between extant and target systems is summarised in Figure 9-2.

At a minimum, it is recommended that the extant *current* system should be analysed to support a better conceptualisation of the target system. A number of extant *related* and *partial* systems may then be analysed (in that order) to augment the design data base. It should be emphasised that the selection and analysis of extant systems is iterative, e.g. the analyst may decide to analyse other extant

systems during the course of the ESSA and other JSD*(HF) stages (see Generalised Task Model Stage later). The procedures for selecting appropriate extant systems for analysis are summarised below.

Figure 9-2 : Extant System Categories Assumed by the JSD*(HF) Method

System Category	Organisation Status	Domain Status
Extant Current System	Client	Same or very similar to target system
Extant Related System	Client or Other	Similar to target system
Extant Partial System	Client or Other	Similarities only at sub-task level

Procedures for selecting extant systems

1. Consult the statement of requirements and other sources (such as transcripts of interviews with task performers, system manuals, etc.) for information on key target system characteristics such as the following :

(a) the domain of application
(b) technological constraints
(c) client-specified task constraints
(d) system performance criteria
(e) user characteristics
(f) environmental factors
These characteristics are then used to identify extant systems for analysis.

2. Abstract and generalise key target system characteristics. In particular, identify salient features of the domain of application and task. Appropriate characteristics of the extant current system may also be incorporated.

3. From the generalised set, particular sub-sets may then be selected. Generic categories of extant systems are thus defined, i.e. extant systems which fall within the categories represent potential candidates for analysis. Such generic categories need not include all characteristics of the target system. However, the criteria for selecting a particular set should be made explicit. These criteria are largely situation-specific, e.g. they are determined by particular information requirements, e.g. on critical or problematic aspects of the target task (see (4) below).

4. Select and record a list of extant systems for analysis. The number of extant systems selected is influenced by situational factors such as :

(a) target task characteristics, e.g. task criticality, frequency and difficulty;
(b) resources available for analysis, e.g. availability of current system personnel for interview; time constraints; etc.
(c) target domain characteristics, e.g. similarities with the current system; well- or ill-defined system; designer familiarity with the domain; etc.

(b) Elicitation of Task Information

Having selected a number of extant systems, relevant design information is elicited to support HF analysis and later generation of ESSA Stage products. Information elicitation is facilitated by a number of 'off-the shelf' techniques such as interviews, observational studies, concurrent and retrospective protocols, critical incident analysis, questionnaires, literature survey, etc. These techniques will not be described since they should be familiar to target users of the method. As an example, procedures for task performer interviews (which would be conducted in most cases) are described overleaf.[8] The reader is referred to Diaper (1989a, b) for an account of other elicitation techniques. Suffice it to say that an elicitation technique that meets the requirements of the analysis should be selected.

(c) Derivation of ESSA Stage Products

ESSA Stage products[9] that may be derived following extant systems analysis comprise the following :

(1) Extant Task Description (TD(ext)) : a device-dependent description of the user's task for an extant (ext) system;
(2) Extant Generalised Task Model (GTM(ext)) : a 'device-independent' description of the user's task for an extant (ext) system;
(3) Extant System Task Model (STM(ext)) and User Task Model (UTM(ext)) : device-independent descriptions of the user's on-line (i.e. computer supported) and off-line (i.e. manual) tasks for an extant (ext) system;
(4) Extant Interaction Task Model (ITM(ext)) : a description of device-level interactions currently required to perform the on-line task;

---

[8] Part of these interview procedures are attributed to Silcock who joined the RARDE project (on which the PhD is based) in its final year. Guided by the present author (the project leader), Silcock's contributions (cited as appropriate) comprise a review of task analysis and elicitation methods, and part specification of secondary design activities of the JSD*(HF) method.

[9] Acronyms for these product will be used henceforth.

Procedures for task performer interviews

1. Conduct interviews. To facilitate information elicitation :
(a) use graphical representations during the interview. For example, pertinent design information may be uncovered by asking task performers to describe the task with respect to graphical representations of the device. The elicited descriptions may also be represented graphically and presented later to task performers for confirmation. The graphical notation used would depend on the information being described, e.g. tree diagrams may be used for describing organisational structure, while JSD structured diagrams, tree diagrams and flowcharts may be appropriate for task description.
(b) use 'how' and 'why' questions during the interview. Previous research suggests that answers to 'why' and 'how' questions are usually related to superordinate and subordinate goals respectively. The use of such questions may facilitate subsequent construction of task 'hierarchies' from interview transcripts.

2. For each task performer, transcribe audio and visual records of the interview (if any).

3. Analyse the interview transcripts. Design information that supports the generation of ESSA Stage products is extracted from the transcripts as follows :
(a) pertinent phrases in the transcripts should be highlighted and summarised;
(b) answers to 'how' and 'why' questions are examined to extract the structure of the task;
(c) relations between domain objects should be noted, e.g. composite objects, task groupings, etc. The information may be used to define the domain of the target system;
(d) statements describing user needs, problems and possible solutions should be noted. Such information supports the derivation of an enhanced statement of requirements.

4. Re-describe the information using notations of corresponding stage-wise products of the JSD(HF) method. The objective is to facilitate later transformation of the information into the scope and format of later JSD*(HF) design products.*

(5) Extant Interface Model (IM(ext)) : a description of the appearance and behaviours of bespoke objects of the extant user interface, including variant objects of the chosen in-house style or user interface environment (if any);

(6) Extant Display Design (DD(ext)) descriptions : of static and dynamic characteristics of extant displays (including dialogue and error messages), e.g. screen composition, layout and actuation with respect to the user's task;

(7) Extant Domain of Design Discourse (DoDD(ext)) : a semantic net description of the application domain of an extant system;

(8) Extant Statement of User Needs (SUN(ext)) : a summary of user problems and needs for an extant system.

These products support later assessments for recruiting particular extant designs to the target system. To this end, *extant* system descriptions are processed into the

scope and format of corresponding products derived at later stages of the method. In other words, extant and target system descriptions[10] are 'mirror images' of one another, e.g. ITM(ext) and ITM(y); STM(ext) and STM(y); etc. The uptake of extant system descriptions at later design stages is thus facilitated. It should be noted, however, that in many cases the full complement of ESSA Stage products need not be derived. In particular, ESSA Stage analysis may be terminated following the derivation of a GTM(ext) description if the information captured is considered sufficient for target system design. Thus, a HF designer should assess whether the derivation of a wider range of ESSA Stage products would benefit target system design.[11]

A selective review of ESSA Stage products follows. A complete case-study illustration of the products is unnecessary since extant system descriptions are 'mirrored' by target system descriptions. Thus, the reader should refer to either Chapters Ten and Eleven for case-study illustrations of target system descriptions, or to Annex D for a more extensive account of ESSA Stage products.

(1) Extant Task Description (TD(ext))

TD(ext) is a device-dependent description of the user's task. Its level of decomposition is determined by the purpose of the analysis. For instance, a low level description (e.g. to the keystroke level) may be derived for an extant *current* system (i.e. a system that operates in the same domain as the target system) designed using the same user interface environment (if any). Alternatively, a higher level description of TD(ext) may be derived when extant systems less closely related to the target system domain are analysed, e.g. extant *related* systems. In such cases, the description needs to support comparisons between the conceptual designs of extant systems and target system requirements.

---

[10] JSD*(HF) products corresponding to the target system are denoted by the following scheme : <JSD*(HF) stage name> <(y)>, e.g. GTM(y).

[11] Thus, the rationale for deriving a particular range of ESSA Stage products should be documented.

TD(ext) is described using JSD* Structured Diagram Notation (see Annex A). The description is supported by an information table that provides a textual account of important task design features.

Procedures for deriving a TD(ext) description are summarised below.

Procedures for deriving TD(ext)

1. Take as input information that has been elicited from task performers, manuals, etc. Identify the super-ordinate task (and task goal), and decompose them into sub-tasks (and sub-goals). Note the sequence, frequency and conditions that control the execution of each set of sub-tasks (these will be represented in TD(ext) using JSD structured diagram notation). The objective is to derive a comprehensive description of task components to support subsequent task analysis and design.*

2. Continue the decomposition until a satisfactory level of description is derived. The point at which decomposition is terminated will depend on the purpose of the analysis. For example, if target system design involves re-designing the extant current system using similar technology, then it may be informative to continue task decomposition to the keystroke level. In contrast, if target system design is instigated by the use of substitute technology such a level of decomposition would not be warranted. At a minimum, sub-tasks of the TD(ext) description should detail how superordinate task goals may be fulfilled, e.g. it should describe start- and end-points of major tasks and define pre-requisites for the fulfilment of set goals.

3. Analyse task performance from a HF perspective. Particular attention should be given to:
(a) user problems and needs as well as positive extant design features that may contribute later to target system design;
(b) the rationale underlying extant designs. The information would support later assessments of the efficacy of existing design features.

4. Record the derived information using JSD structured diagrams and include textual comments in an accompanying table as necessary. Note that :*
(a) the table structure may vary according to the needs of the structured diagram. In most cases, it should include a column each for elaborating boxes of a structured diagram and for noting HF comments on particular task design features;
(b) table items should be recorded in the sequence by which structured diagrams are read, i.e. from top to bottom and then from left to right;
(c) complete elaboration of all boxes of a structured diagram in the accompanying table is unnecessary since the objective is to clarify complicated parts of the diagram and not exhaustive documentation. In particular, the selective elaboration of structured diagram boxes is intended to balance the requirements for adequate documentation and economy of effort (limited project resources).

Generification and TD(ext)

On most occasions, task descriptions for an extant current system would be elicited from various information sources. To derive a single TD(ext) description, generification techniques have to be applied to identify and remove subjective variation across descriptions elicited from each of the sources.

Generification may be considered a special instance of abstraction (see Annex B). Its main purpose is to identify a superordinate classification (generic descriptor) for a set of two or more entities based on shared attributes. Thus, generification may be applied to extract a single TD(ext) description from a set of related extant tasks. In particular, subjective variation across individual task descriptions are removed by defining a generic descriptor for task elements that share common attributes. Details of the generification procedure are described below (Silcock and Lim, 1990a).[12]

Procedures for generifying extant task descriptions

1. Collate a list of all objects and actions from the information elicited.

2. Reduce the set by listing each object and action once and once only.

3. Associate all similar terms using one of the following techniques :

Technique 1: The designer associates a particular term with other similar terms iteratively by expressing the original task description in terms of an alternative object or action. If the alternative description is considered 'adequate', then the terms may be considered similar.

Technique 2: Task performers are asked to sort object and action cards into different groups on the basis of task relevant criteria.

4. (Subjectively) assign a common or generic label to each group of 'like' terms.

5. Validate the generic description derived by asking task performers to assign a generic label (from (4) above) to each item in the original list of objects and actions. If a specific item can not be located satisfactorily, task performers are free to supply an alternative label. Thus, suitable generic entities are identified iteratively.

6. Giving due consideration to the original information, construct a generic task description using generic entities where appropriate.

7. Validate the generic task description with task performers by comparing the generic description with the account elicited from task performers.

[12] See Footnote 8.

Two extant systems were analysed in the network security management case-study, namely an extant *related* system in use at University College London and an extant *partial* system represented by the PC security application, MacPassword.™ The *target* system comprises a hypothetical system to be implemented at the Royal Armament Research and Development Establishment (simulated client). It should be noted that the extant *current* system was not analysed owing to military security and the unavailability of security staff at the client organisation for interview.

More needs to be said about the extant systems selected for analysis. The network security management system at University College London was selected because it shares the same domain as the target system. Thus, information derived from its analysis may support the generation of a conceptual design for the target system. For instance, security management tasks in both systems comprise accessing the computer followed by the detection, identification and response to security breaches. Similarly, MacPassword™ is selected as its domain (i.e. *personal computer* security management) is *partially related* to *network* security management. Its selection for analysis was motivated by the potential recruitment of its low level design features to the target system.

Examples of the TD(ext) descriptions derived for the extant system at University College London Computer Centre (TD(UCLCC)) and the MacPassword™ application (TD(MPASS)) are shown in Figures 9-3 and 9-4 respectively. Information tables for each of these descriptions are also shown.

To illustrate how extant system analysis may contribute to target system design, part of the security management task is examined. Specifically, the task concerns the identification and response to failed log-on events arising from illegal or incorrect password inputs. These events are important since they may signal attempted access by a hacker. In this respect, the TD(UCLCC) description indicates that the extant system did not provide a facility to alert the network manager to such events. Thus, these security breaches can only be uncovered by manually searching through volumes of computer logs or network user reports. To rectify these design

Figure 9-3 : Task Description for Network Security Management at University College London Computer Centre (TD(UCLCC)) -- Page 1

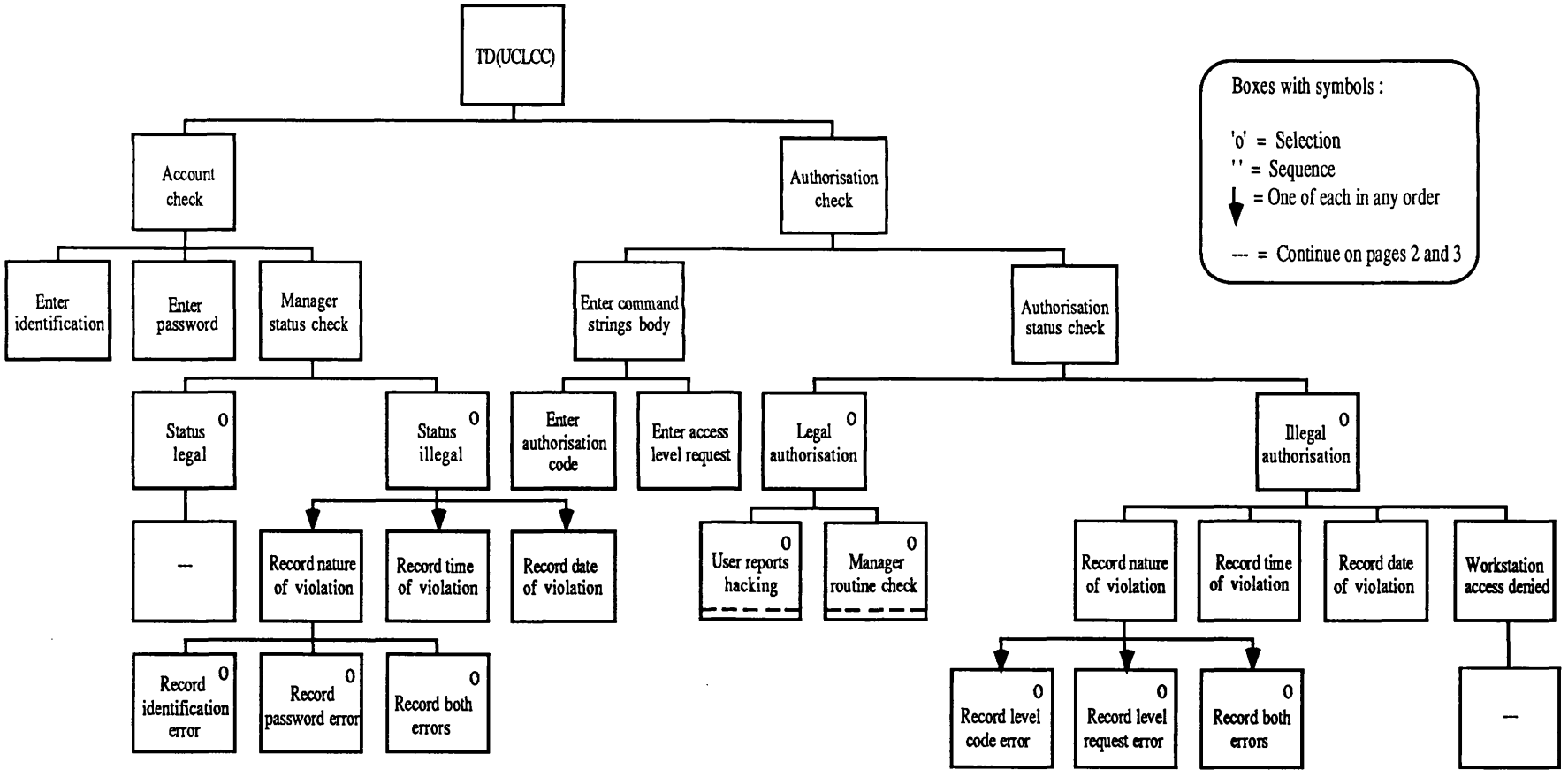


Figure 9-3 : Task Description for Network Security Management at University College London Computer Centre (TD(UCLCC)) -- Page 2

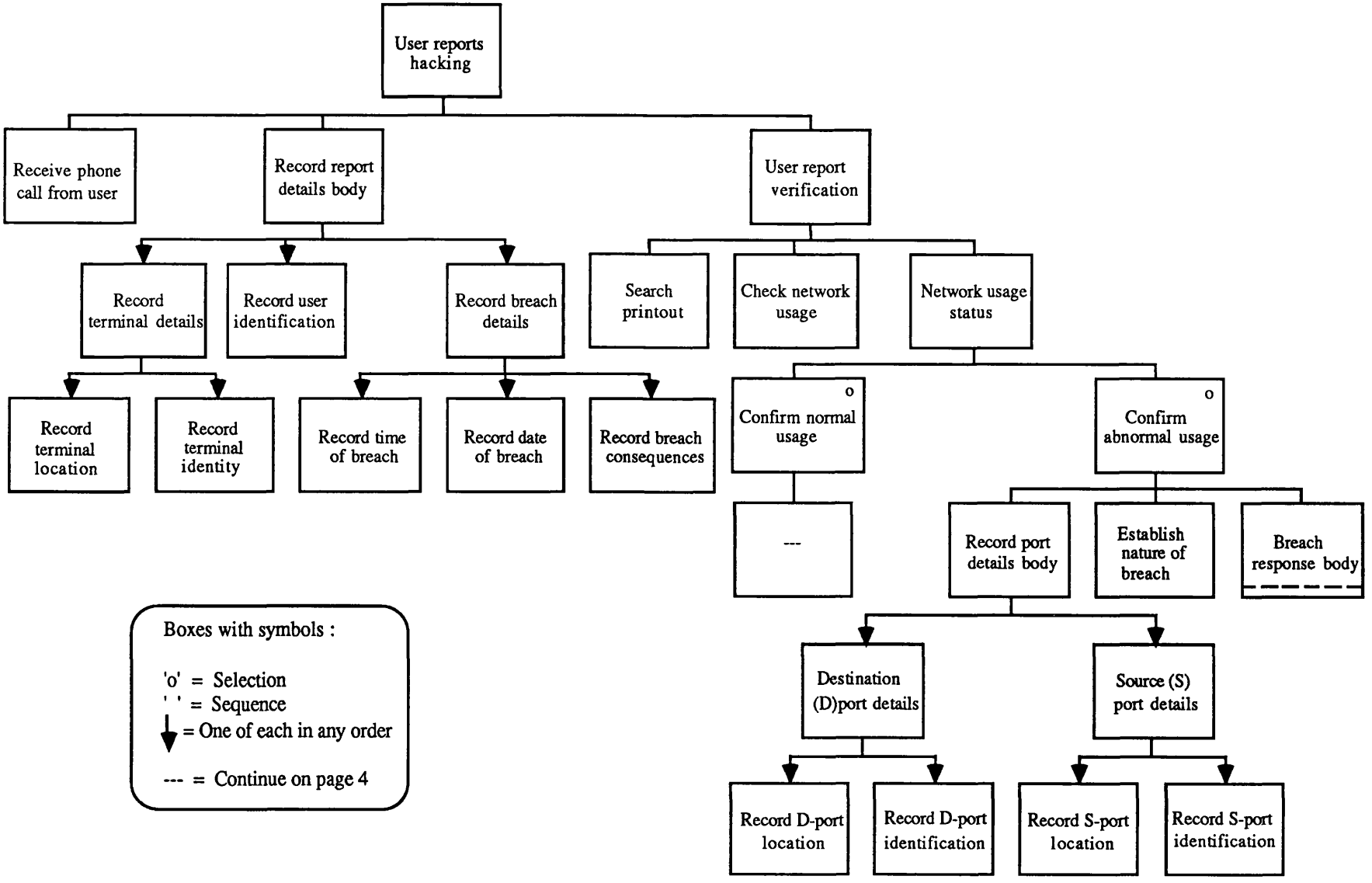


Figure 9-3 : Task Description for Network Security Management at University College London Computer Centre (TD(UCLCC)) -- Page 3

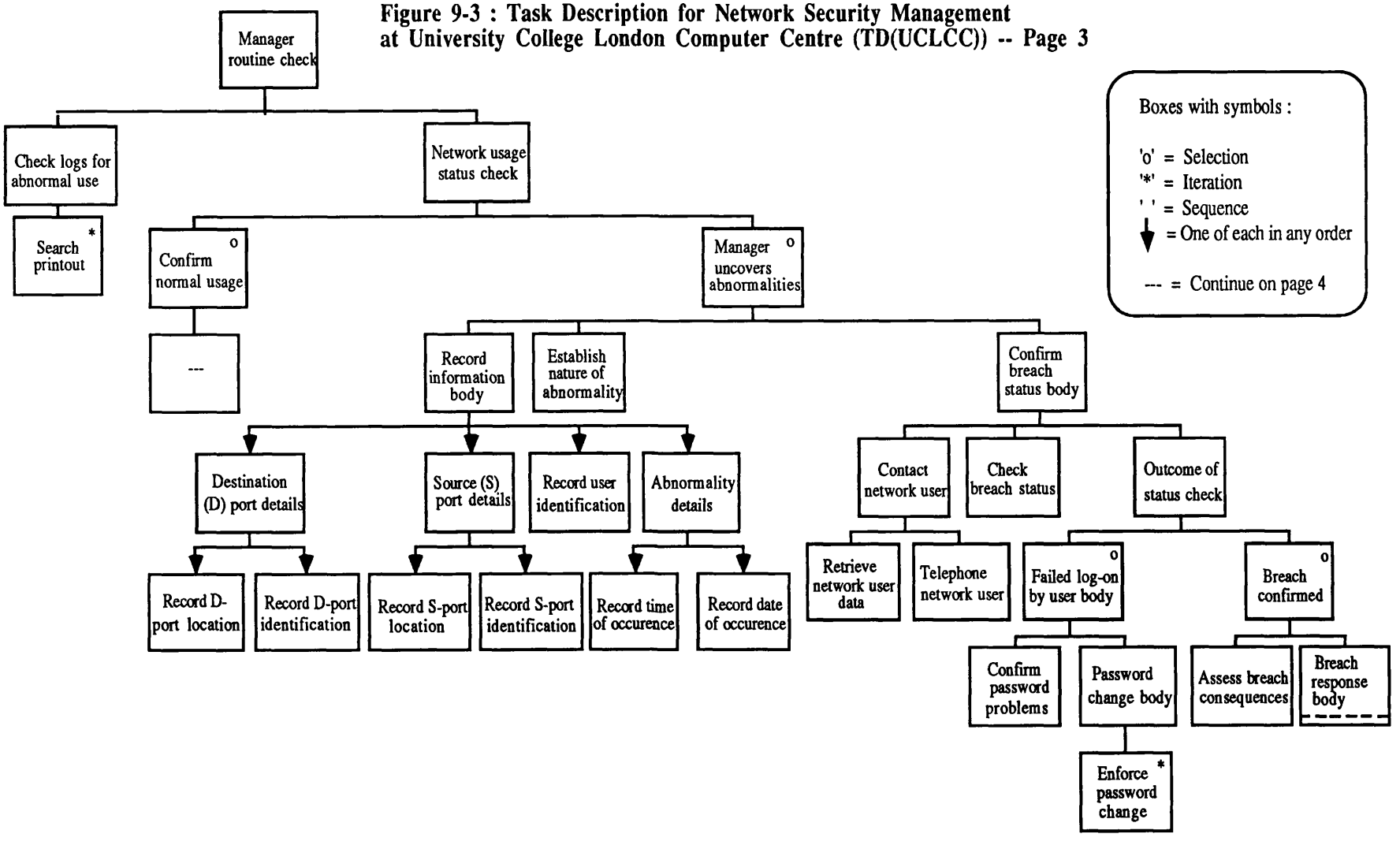
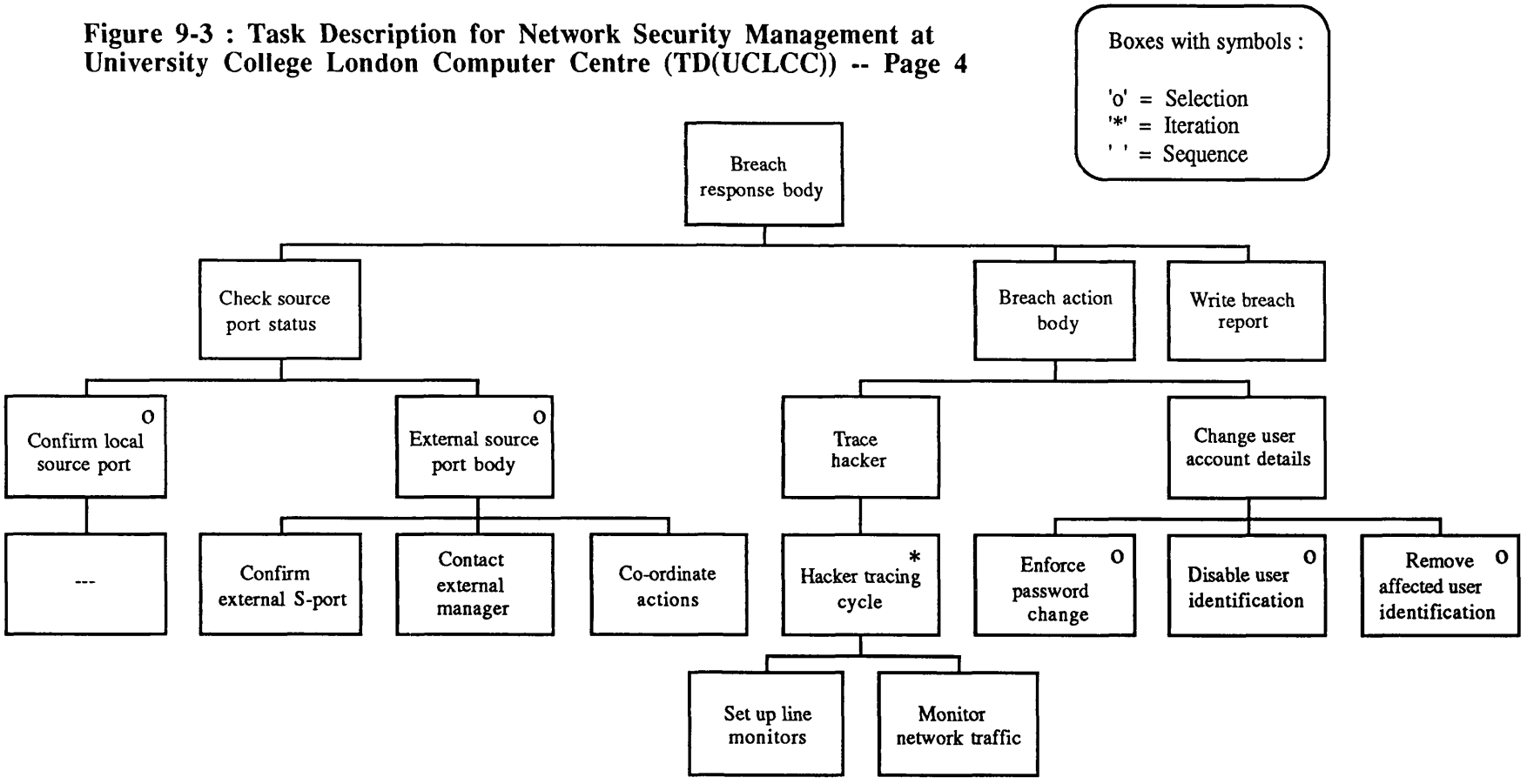


Figure 9-3 : Task Description for Network Security Management at University College London Computer Centre (TD(UCLCC)) -- Page 4



TD(UCLCC)Table -- Page 1

Name	Description	Observation	Design Implication	Speculation
Account check Authorisation check	The network manager was required to clear two checks, namely account and authorisation checks, before access to the network management workstation was permitted. Violation of either of these checks was recorded by the computer.		The access checking system was considered necessary for UCLCC as room security can not be ensured sufficiently. Also a facility for remote access to the workstation was in place. Thus, an authorisation checking scheme was required.	The authorisation check may not be necessary in a more secure environment, e.g. Fort Halstead, RARDE (the target system site).
User reports hacking	A network user may contact the network manager to report a security breach (e.g. an illegal log-on). In most cases, the network manager was expected to act on the report immediately. This action involves searching through the system log (print-outs) to verify details of the user report.	Report details are recorded by the network manager as off-line logs.	Asynchronous communication should be supported.	An electronic-mail facility may support the task better.
Manager routine check	The network manager may occasionally scan logged information for evidence of possible security breaches (off-line search). Details of possible security breaches were recorded and subsequently verified with the user (both off-line tasks).	There was no indication that the structure, content and layout of computer logs were designed to support this task, except line indentations and spacing to denote separate log entries.	Logs should clearly indicate occurrences of failed log-ons. On-line functions should be provided to support the task of scanning log-on times for potential illegal log-ons.	On-line support functions may provide specific information retrieval such as the following : (i) failed log-ons and specific user identification. The output should show time, source and destination addresses; (ii) time intervals and user identifications. The output should show source and destination addresses.
		Breaches were typically discovered after the event	On-line alert should be provided for failed and illegal log-ons.	Auditory and visual alarms may be provided in response to failed log-on, with post-event summaries in case the workstation is unmanned at that time.

Name	Description	Observation	Design Implication	Speculation
Check logs for abnormal use	The log-on address was located by the network manager based on information on the security breach event and user identification. The address may indicate the physical location of the hacker (e.g. from a specific room in the college or by remote access from other networks).	There was no indication that the structure, content and layout of computer logs were designed to support this task.	On-line information retrieval facilities should be provided to support easy access to user and network details.	Support function should show network use details on input of specific user name and time.
Confirm breach status body	The network manager had to contact the user to ascertain that a security breach has actually occurred. If the abnormal use event is due to a legal user, no further action would be taken. Alternatively, a password change may be enforced so that the user may choose a more easily remembered password.	It may be difficult to identify specific users if an identification is shared by a number of individuals.	Unique user identification is required.	
Assess breach consequences	The network manager may search through system logs to determine the consequences of the breach.	Reported to be a slow and demoralising process.		
External source port body	If a hacker has logged onto the network via another network, the network manager may attempt to identify the remote source address and so acquire information on the hacker's location and identity. The undertaking would involve contacting managers of other networks to arrange for the information to be forwarded. (Note : this could be done by telephone).	It may be difficult to obtain the desired information.	The network manager should have easy access to remote personnel as far as possible. Asynchronous communication facilities would be advantageous.	Electronic-mail may be provided (depending on facilities of remote networks).
Trace hacker	The network manager may set up line monitors to detect hacker activity in real-time, and hence trace its source.	Little information was available on this task except that it was a labour intensive undertaking.		

Name	Description	Observation	Design Implication	Speculation
Enforce password change	The network manager may enforce a password change on a user in the following circumstances : (a) to rectify a security breach attributed to an illegal log-on; (b) to enable the selection of a more appropriate password in response to failed log-ons by the user.			
Disable user identification	The network manager may temporarily disable a user identification (e.g. if the user cannot be contacted to confirm whether hacking has taken place, or be notified that a change of password has to be enforced).			
Remove user identification	The network manager may permanently remove the affected user identification after contacting the user concerned. A replacement account may be set up if necessary.			
Write breach report	The network manager may require a record of such events in the long term so that any pattern in hacking attempts by the same individual may be revealed.			

Figure 9-4 : Task Description for PC Security Management for the MacPassword™ Application (TD(MPASS)) -- Page 1

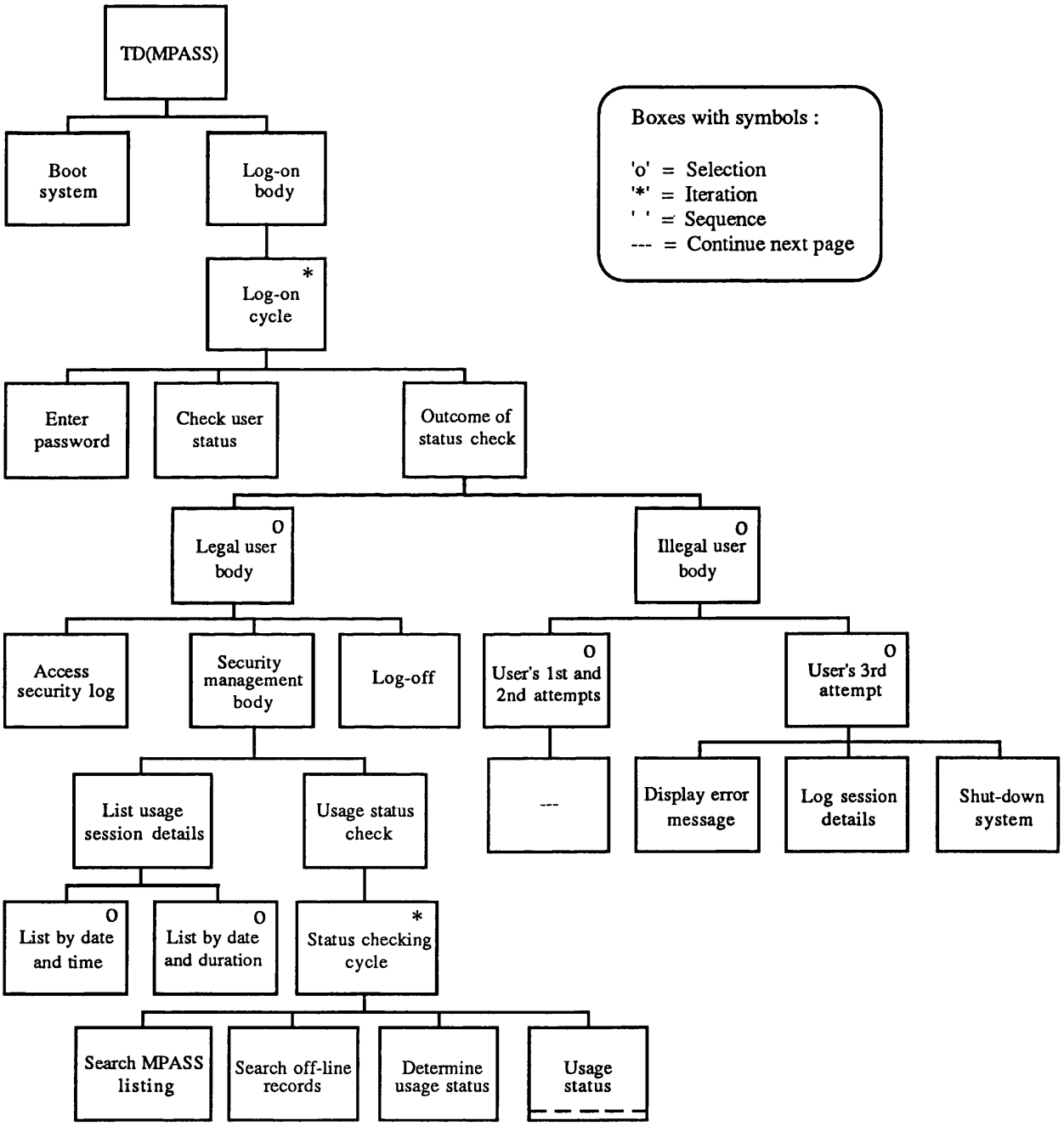
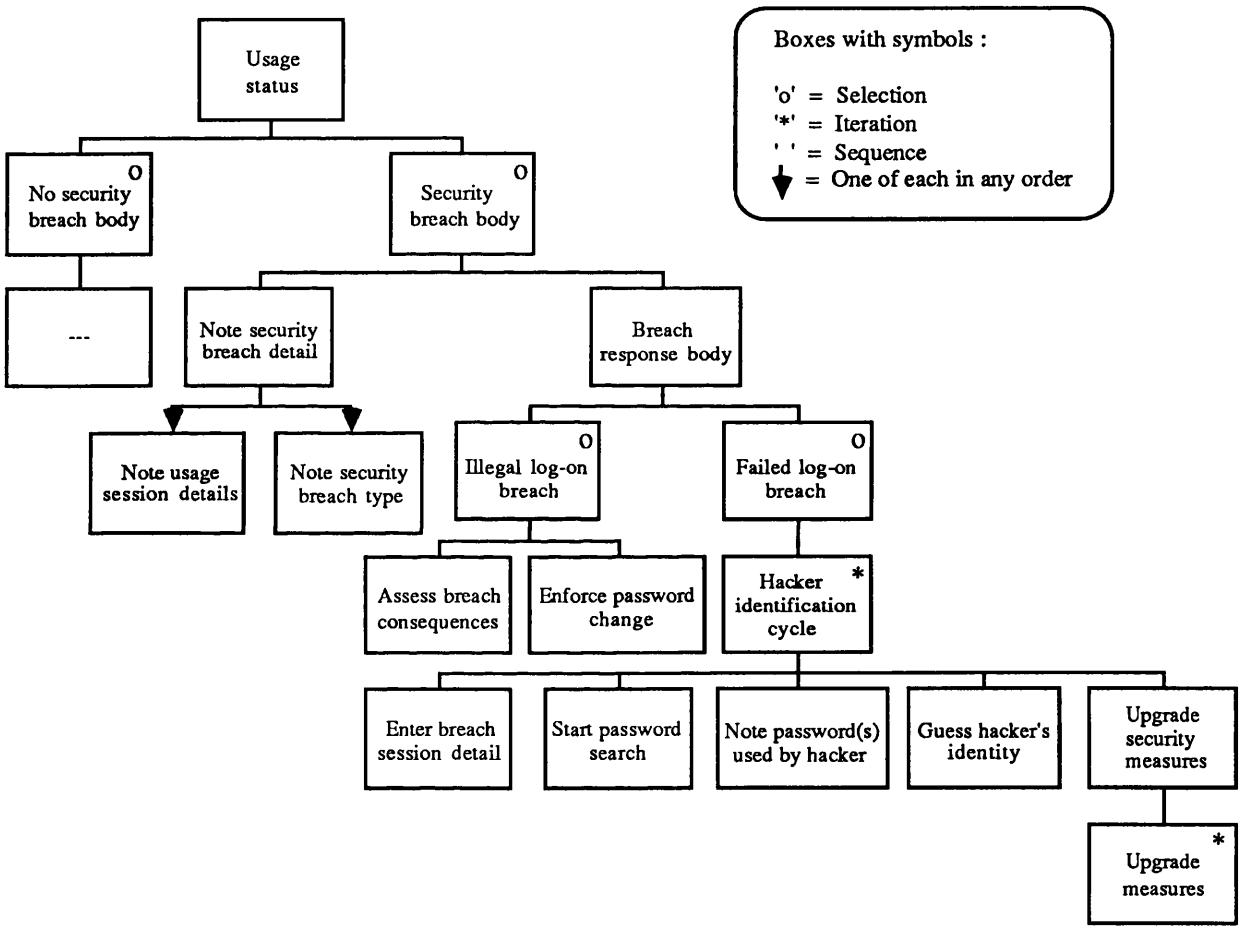


Figure 9-4 : Task Description for PC Security Management for the MacPassword™ Application (TD(MPASS)) -- Page 2



TD(MPASS) Table -- Page 1

Name	Description	Observation	Design Implication	Speculation
Boot system	The user was required to boot the computer to access the system.			
Enter password	A password is entered by the user for verification by the computer.	No user identification was required. Less secure system.		
Legal user status body	A correct password was specified by the user. Once logged on, the user may carry out transactions for any of the system's services.	Easy access to system logs; and other security sensitive information.	Need a stronger distinction between security levels following user log on.	Define unique passwords for separate facilities to provide additional security.
Access security log	The security log was written to a file in the system folder.	The log was the only means of identifying hacking activity. It was thus particularly important to make logged information inaccessible to hackers.	Need some means of protecting logged information and for facilitating data search.	Define unique password for accessing system logs.
Security management body	The owner must access the database for log-on, log-off times and dates, and check these against known systemusage. If particular usage activities cannot be accounted for, the session details are noted. Such irregularities may indicate hacker activity. Further action may then be taken.	May become cumbersome if many log-ons were involved over a period of time. A separate and comprehensive record of system usage was normally not kept by the owner to support usage comparison. Thus, some log-ons may be difficult to verify.	Need support functions for logging time, date and password used during failed log-on events. Need to support the search process by providing facilities for the direct retrieval of specified periods of non-use.	A function to search computer usage within a specified time interval. Functions to retrieve logged on and off events for specified time periods.
Enforce password change	The password may be changed following hacker activity.	Changes should be effected regularly, i.e. not only after hacking activity has been detected.	Provide support for password changes.	Support password changes in response to failed log-ons and at regular intervals.

TD(MPASS) Table -- Page 2

Name	Description	Observation	Design Implications	Speculation
Failed log-on breach	Failed log-ons were recorded. The owner must identify such occurrences from the log. Other information of interest include the date and time of the event and the password used by the hacker.	Date and time information may provide useful clues to a hacker's identity. A password record may be useful in this respect.	Need to contextualise the information recorded to the task of hacker identification.	Ensure passwords used in failed log-ons are logged.
User's 3rd attempt	The session was terminated only after the third attempt.	After re-booting the computer, the user was free to try again.	The security system should limit multiple attempts by re-booting.	Lock out after failed log-on, or include a time out facility for repeated access following a re-boot.
Log session details	A record of system access events was kept. The information recorded include date and time of owner and guest log-on; failed log-on; and password(s) used.			

inadequacies, computer supports suggested by the analysis include automatic alerts to failed log-on events and information collation functions. These suggestions were substantiated by the TD(MPASS) description since similar requirements were identified, i.e. computer owners need to consult computer logs for evidence of failed log on events. In this respect, the analysis highlighted a potentially useful design feature in MacPassword,™ namely computerised recording of passwords used in such events. Such records may provide clues to the identity of the hacker and may help to distinguish between illegal and incorrect password inputs, i.e. password mis-keys may be differentiated from hacking attempts.

(b) Extant Generalised Task Model (GTM(ext))

The objective of a GTM(ext) description is to support conceptual assessments of extant system(s) designs relative to target system requirements. To this end, GTM(ext)s are derived for all extant systems analysed at the ESSA Stage. Extant designs that may be ported to the target system are thus exposed.

GTM(ext) is derived by eliminating a proportion of the detail in TD(ext) that is specific to a device. In other words, TD(ext) is a device-dependent description while GTM(ext) tends towards device-independence. Thus, the logic underlying a specific task is brought out by abstraction. Despite their tendency towards device-independence, GTM(ext) descriptions should retain sufficient information about relevant extant design features, e.g. design rationale. In particular, the level of GTM(ext) description should not be too high that design features of interest are lost (see rules of thumb for GTM(ext) derivation overleaf).

As with TD(ext), GTM(ext) is described using structured diagrams and an information table. Procedures for deriving GTM(ext) descriptions are summarised on the next page.

Procedures for deriving GTM(ext)

Starting at the top of the TD(ext) description, work through each node as follows :

1. *summarise the semantics of the task (and its sub-tasks) in 'device-independent' terms to reveal the underlying logic of the particular node. This process is supported by by GTM(y) since it constitutes a 'device-independent' structure for the target system. Specifically, GTM(y) and the statement of requirements provide guidance on the level of GTM(ext) description. In general, the more similar the extant and target systems are to one another, the more device dependent details should be retained in the GTM(ext) description (see rules of thumb below).*
2. *continue abstracting each node down the TD(ext) hierarchy. This process may become more difficult at lower levels since they are, by nature, likely to be more device specific (e.g. task inputs and outputs). Note that GTM(ext) descriptions should relate to salient characteristics of the target system so that later comparisons are facilitated.*
3. *record GTM(ext) descriptions using JSD* structured diagram notation, and include additional notes in a supporting table as appropriate.*

Rules of thumb for GTM(ext) derivation

1. *The level of GTM(ext) description should be high enough to facilitate comparison between extant and target systems.*
2. *The level of GTM(ext) description should be low enough to capture sufficient information of interest. Thus, a completely device independent description is not always desirable since information on the relationship between particular task characteristics and device design would be lost.*
3. *A one-to-one mapping between device-dependent and device-independent descriptions is unlikely, particularly at lower levels of description. Thus, it may not be possible to abstract a TD(ext) node directly in 'device independent' terms. In such cases, it may be necessary to combine the node with an adjacent node(s) and consider their abstraction into one device independent term.*

Case-Study Illustration of GTM(ext)

A case-study example of a GTM(ext) description is shown in Figure 9-5. The Figure shows that on abstracting the description (i.e. GTM(UCLCC)) from TD(UCLCC) :

(i) lower level details were omitted in favour of a more general description, e.g. the 'Status illegal' sub-node and its leaves (Figure 9-3, Page 1) have been reduced to a single 'Record violation details' leaf (Figure 9-5, Page 1);
(ii) device-specific task information was removed, e.g. the 'Search printout'

leaf (Figure 9-3, Page 3);

(iii) its overall task structure was largely carried over to the GTM(UCLCC) description;

(iv) details outside the scope of target system design were excluded. Thus,

Figure 9-5 : Generalised Task Model Description for Network Security Management at University College London Computer Centre (GTM(UCLCC)) -- Page 1

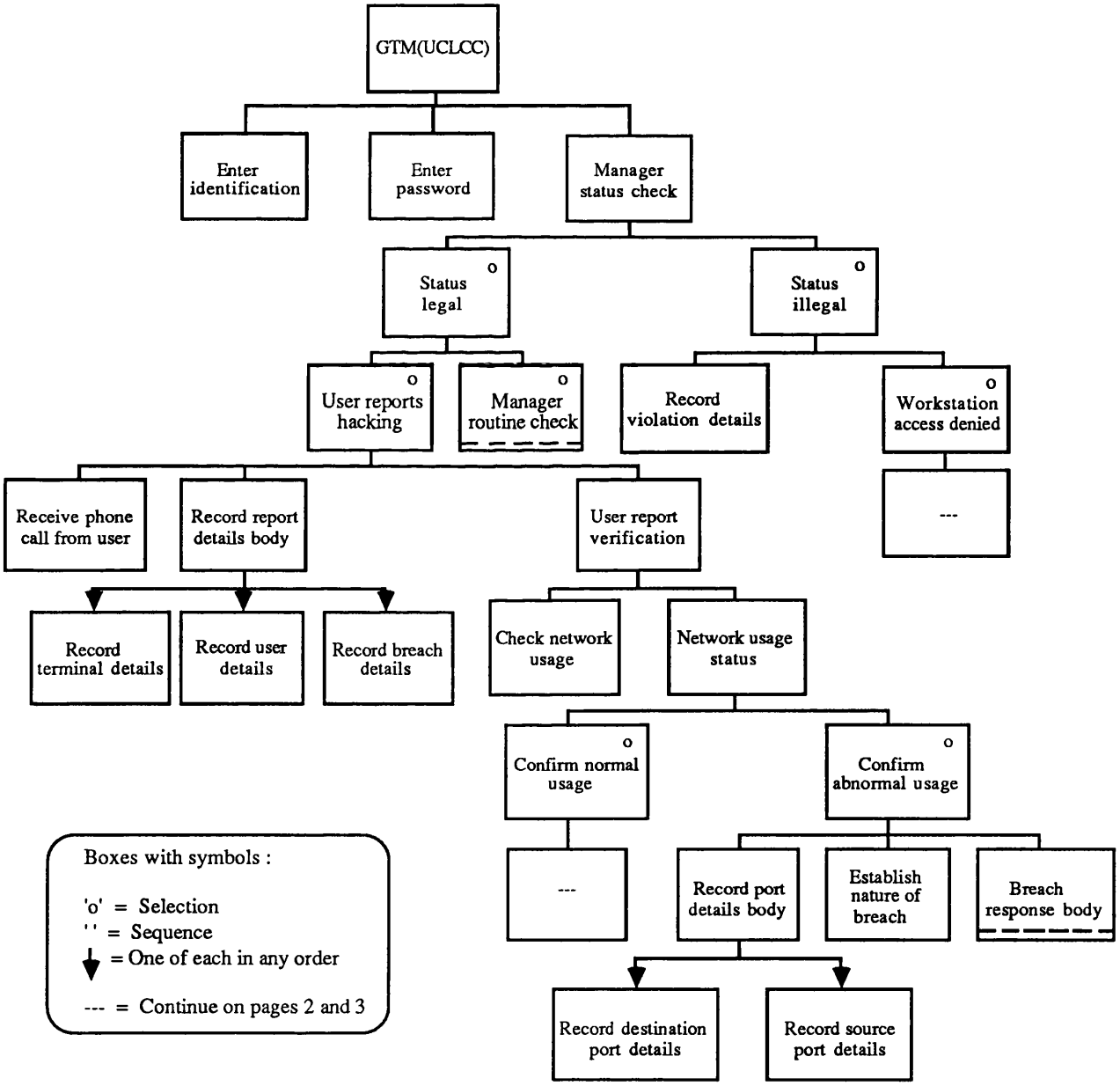


Figure 9-5 : Generalised Task Model Description for Network Security Management at University College London Computer Centre (GTM(UCLCC)) -- Page 2

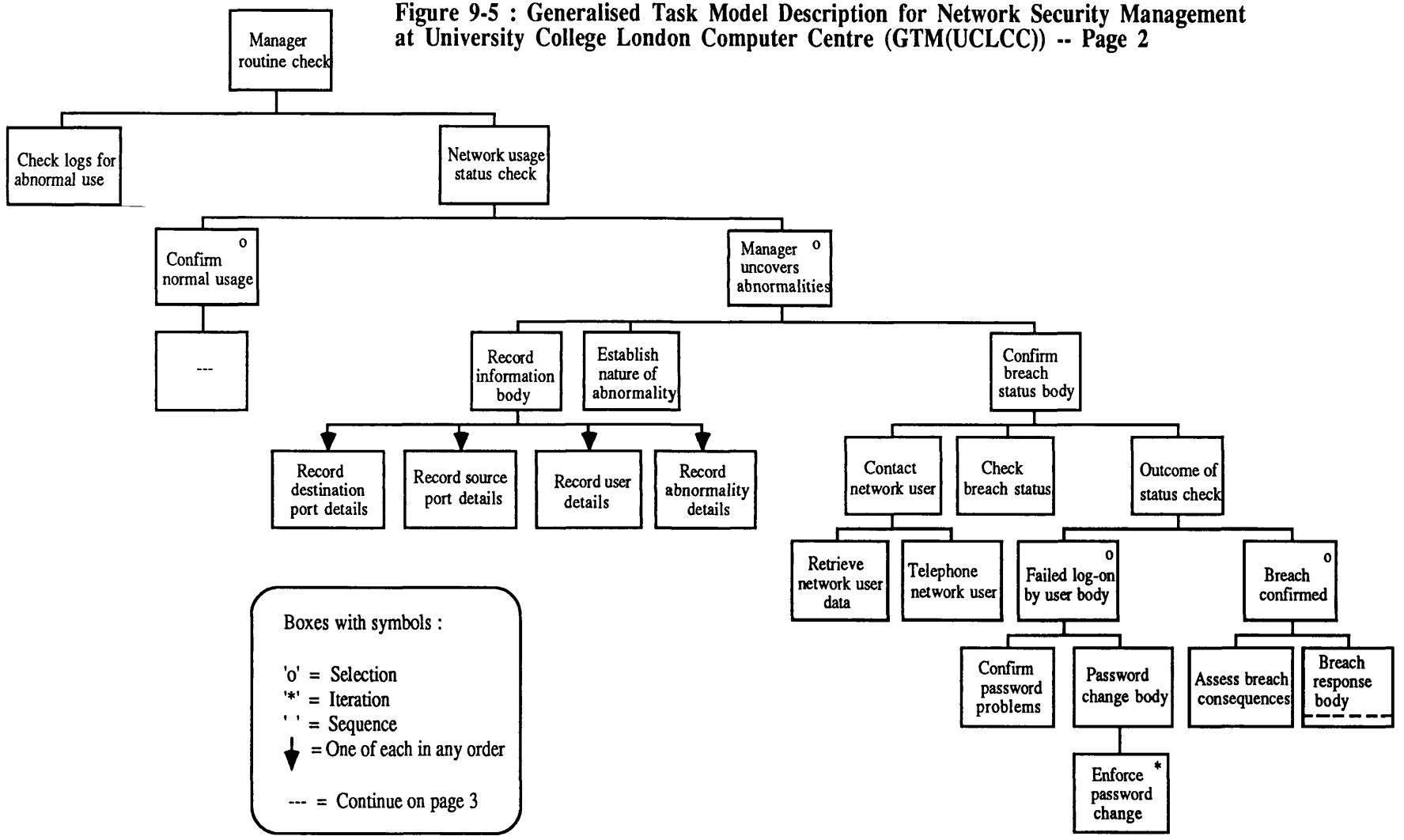


Figure 9-5 : Generalised Task Model Description for Network Security Management at University College London Computer Centre (GTM(UCLCC)) -- Page 3

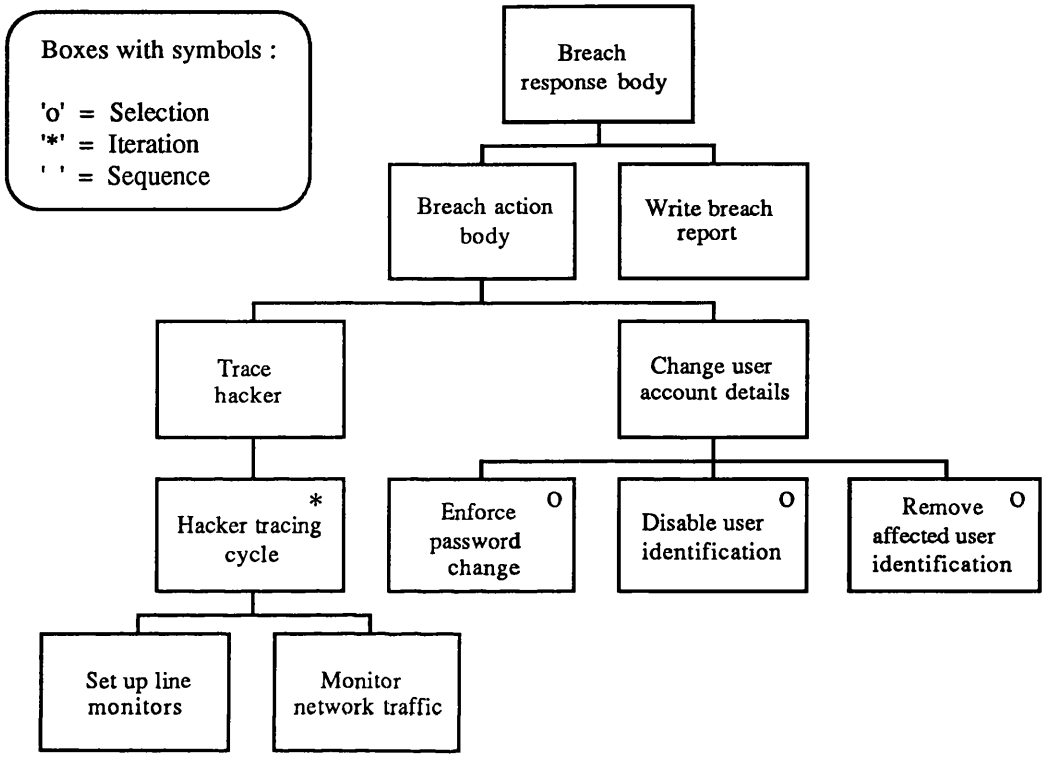


Figure 9-5 may be considered a GTM"(UCLCC) description, i.e. a subset of a complete GTM(UCLCC) description. In particular, the 'Check source port status' sub-node in Figure 9-3, Page 4, was omitted since the scope of the target system did not include connections with external networks.

A design information table was not derived in this instance since TD(UCLCC) tables provide sufficient support for interpreting the GTM(UCLCC) description.

As for MacPassword™, a GTM(ext) description was not derived since the TD(MPASS) description was adequate to support assessments of the potential porting of its design features to the target system. Further case-study examples of a wider range of ESSA Stage products (derived in specific instances) are provided in Annex D.

A summary of ESSA Stage procedures is provided below.

High level procedures of the ESSA Stage

(i) Examine contractual documents and records of the clients' brief.

(ii) Collate the information to derive an initial statement of requirements.

(iii) Select the current system used by the client.

(iv) Elicit and analyse user, task and general design characteristics of the current system. The scope of elicitation and analysis should relate to the scope of the target system.

(v) Identify further extant systems for analysis if more information is desired. The activities in step (iv) above are thus repeated.

(vi) On the basis of the information derived, generate the requisite range of ESSA Stage products. The products actually generated depend largely on the prevailing design scenario. However, the basic set should include the Task Description and Generalised Task Model for each extant system analysed.

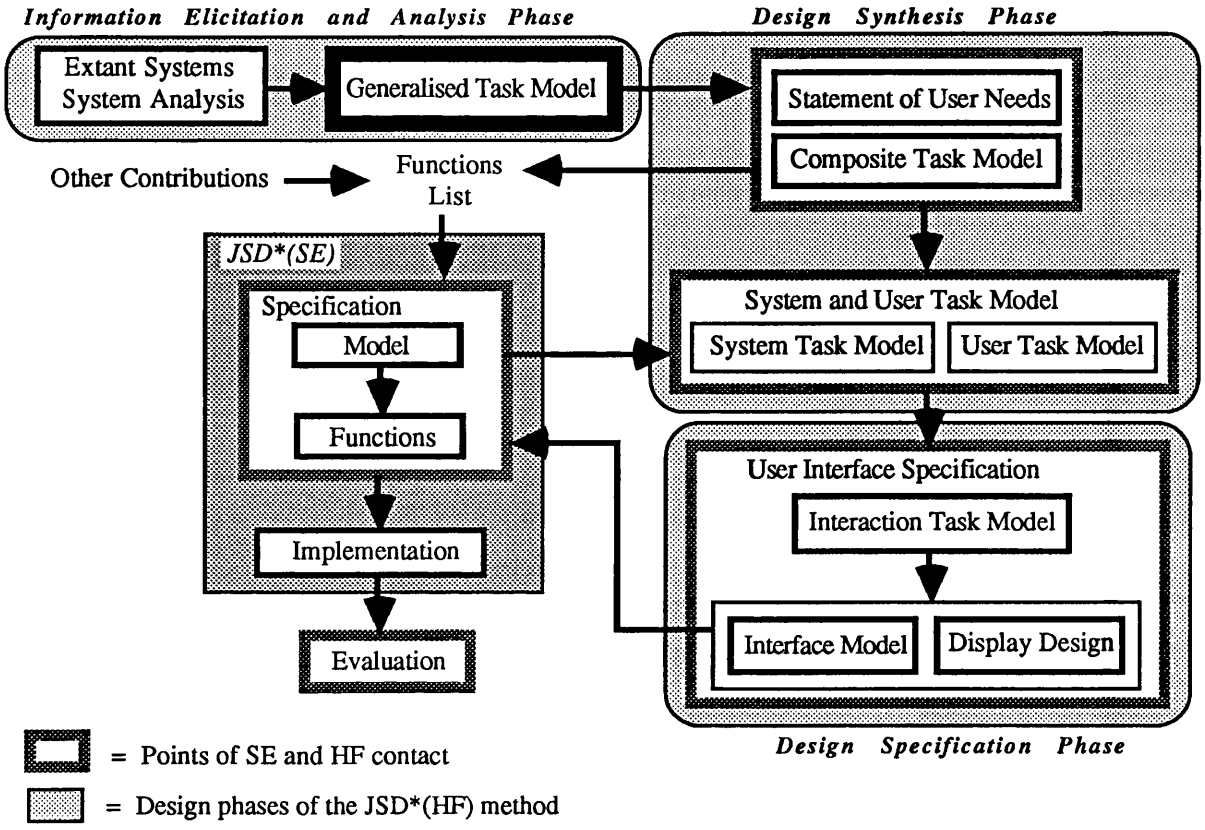
An account of design activities for the next stage of the method (i.e. the Generalised Task Model Stage) follows.

9.2. Generalised Task Model (GTM) Stage

Summary

The second stage of the method, namely the GTM Stage, is concerned with the generation of design descriptions to support the conceptual design of the target system (see Figure overleaf. The present stage is indicated by a box highlighted in bold). Specifically, device independent descriptions[13] are generated to facilitate analytic mapping between appropriate extant design features and target system

[13] The extent of abstraction to device-independence is determined by how dissimilar the characteristics of the extant system are with respect to the target system. The level of description should be high enough to reveal logical aspects of the task (as opposed to the device, e.g. detailed interaction sequences). The final description should also retain sufficient information on extant design features that may be relevant to the design of the target system.



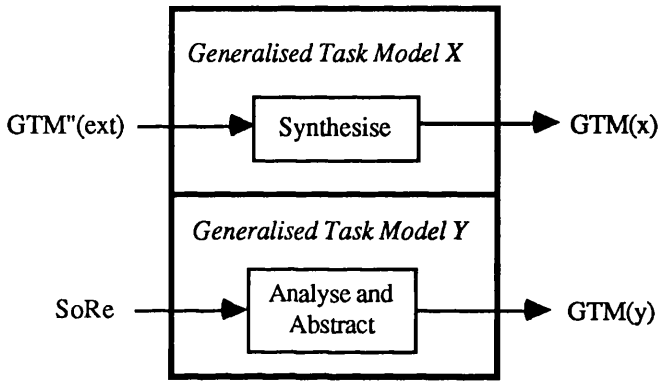
requirements. To this end, two products are derived, namely a generalised *extant task* model (termed GTM(x)) and a generalised *target task* model (termed GTM(y)). These models establish the foundation for recruiting extant system designs and for specifying design extensions in respect of new target system tasks. Thus, an early indication of the training required by target system users may be inferred from GTM(x) and GTM(y) descriptions. Specifically, inferences may be drawn from an assessment of the complexity of the GTM(y) description, and the transfer of learning (both positive and negative) attributed to the extent of porting from the GTM(x) description.

The generalised task models derived are carried forward to the Composite Task Model Stage where appropriate elements are synthesized on the basis of the statement of user needs (see later). In other words, a composite task model is derived by synthesising compatible and complementary sub-sets of GTM(x) and GTM(y).

Detailed Account

An account of the activities for deriving GTM(x) and GTM(y) descriptions follows. As before, italics will be used to highlight the design activities shown in Figure 9-6.

Figure 9-6 : Block Diagram Summary of the Generalised Task Model (GTM) Stage



(ext) = JSD(HF) descriptions of an extant systems (EXT)*
SoRe = Statement of Requirements
(x) = JSD(HF) descriptions of an extant system 'composite' (X)*
(y) = JSD(HF) descriptions of the target system (Y)*

(a) Generalised Task Model of the Target System (GTM(y))

GTM(y) represents the first attempt at defining a conceptual design for the target system. The objective of GTM(y) is to expose new and salient features of the target system and to define the scope and structure of its tasks. Since the description illuminates key characteristics of the target system, GTM(y) also suggests further extant systems for analysis. Thus, GTM(y) provides the basis for initial design development.

To derive a GTM(y) description, the initial statement of requirements is *analysed* and task details are *abstracted* to a conceptual level. Since the statement of

requirements originates from the client and is beyond the control of the method, sufficient information to support the derivation of a comprehensive GTM(y) can not be guaranteed. Thus, on most occasions, GTM(y) must be enhanced within the method to support the definition of a reasonably complete conceptual task for the target system. In this respect, products of extant systems analyses such as the conceptual task description for the current system (i.e. GTM(ext)s), could be consulted. GTM(y) is thus extended by incorporating pertinent sub-sets of GTM(ext) descriptions. Procedures for deriving GTM(y) are described below.

Procedures for deriving GTM(y)

1. *Take as input task information described in the statement of requirements. Temporal and conditional aspects of task execution should also be noted.*
2. *Summarise the task (and sub-tasks) in device independent terms to reveal the logic underlying its design.*
3. *Re-express the GTM(y) description using the JSD* structured diagram notation, and record additional notes in a supporting table.*

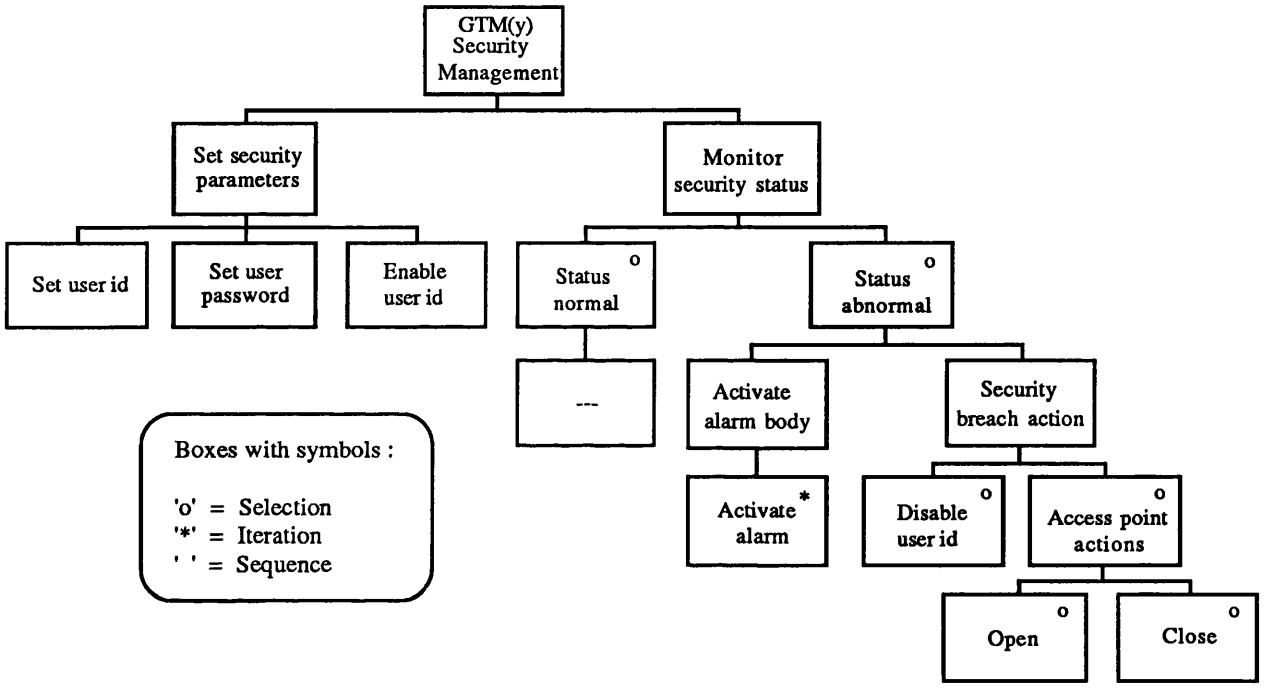
Case-Study Illustration of GTM(y)

A simple case-study example of GTM(y) is shown in Figure 9-7. The description is derived from the statement of requirements as defined by the client organisation (simulated by the Royal Armament Research and Development Establishment). The statement of requirements provides little information on the network manager's task. For instance, it fails to distinguish between different types of security breach and the actions to be taken. Thus, Figure 9-7 indicates that user access is invariably disabled for any security breach.

To rectify such inadequacies, the HF designer may resort to extant systems analysis to generate more detailed task information. Since security staff at the client organisation were unavailable for interview (due to military security), the extant *current* system could not be analysed. Instead, extant related and partial systems were selected for analysis. These extant systems correspond respectively to the

network management system at University College London and the PC security application, MacPassword™.

Figure 9-7 : Generalised Task Model of Network Security Management for the Target System



(b) Generalised Task Model of Extant Systems (GTM(x))

Following the definition of a conceptual task by GTM(y), promising sub-sets of GTM(ext)s may be synthesised into a composite representation termed a GTM(x).[14] Thus, GTM(x) encapsulates conceptual designs of extant systems that could potentially be ported to the target system. To this end, low level details of extant designs corresponding to the selected set are documented.

[14] Synthesis of GTM(ext)s is only necessary if more than one extant system has been analysed, e.g. when related and current extant systems are analysed. In other words, if only the current system is analysed, then GTM(x) = GTM(ext) = generalised task model of the extant current system.

GTM(y) Table

Name	Description	Observation	Design Implication	Speculation
Set security parameters	A number of network security parameters may be controlled by the network manager.	The focus of the client was entirely on network security. Need to arrange additional security measures for the network management workstation.	Need to highlight security measures for accessing the network management workstation.	
	For instance, the network manager can change a user's password, and either enable or disable a user identification (id) at any time. The initial password is allocated by CA2, RARDE. On the first transaction, the user may choose a new password.	A password comprises one alphabetic character followed by seven alphanumeric characters.		
Monitor security status	Network activities (e.g. user log-ons) should be monitored by the network management workstation. On detecting abnormal events (e.g. failed log-ons), the worstation should activate an alarm to alert the network manager.	No suggestion from the client on what form of alerts and subsequent actions by the network manager would be required.		
Access point actions	The network manager should be able to allow or deny access to any network access point (either host or terminal). In particular, the design feature should be capable of disabling all access to selected host machines if required, e.g. to secure sensitive information.			

The procedures for deriving a GTM(x) description are given below.

Procedures for deriving GTM(x)

1. Take as input GTM(ext) descriptions of various extant systems analysed at the ESSA Stage.

2. Compare the GTM(ext) descriptions with target system requirements (a subset of which is represented in GTM(y)). On the basis of earlier HF evaluations, identify aspects of extant tasks that are potentially relevant to target system design. Also note potential influences from other ESSA Stage products, i.e. consider any other (ext) products that have been derived at the latter stage.

3. Identify sub-sets of GTM(ext) descriptions which are compatible. These sub-sets comprise desirable extant task characteristics that may be carried over to the design of the target system.

4. Synthesise the selected subset (i.e. GTM"(ext) descriptions) into a single representation, i.e. GTM(x).

Case-Study Illustration of GTM(x)

An example of GTM(x) for network security management is shown in Figure 9-8. This example was derived by synthesising the GTM(UCLCC) and TD(MPASS) descriptions presented earlier (see Figures 9-4 and 9-5). It can be seen from Figure 9-8 that system access characteristics of MacPassword™ have been recruited (Figure 9-8, Page 1) and synthesised with the security management tasks derived from University College London Computer Centre (see Figure 9-8, Pages 1 to 3. Note that Pages 2 and 3 of the Figure are the same as those of Figure 9-5). No supporting table for the GTM(x) description was considered necessary since its structured diagram nodes have already been detailed in corresponding TD(ext) descriptions.

This account completes an overview of the Generalised Task Model (GTM) Stage. Together with the ESSA Stage, design activities of the Information Elicitation and Analysis Phase are now complete. The conceptual design solutions derived presently are developed further in the Design Synthesis Phase. To this end, the initial statement of requirements needs to be augmented to define an adequate set of

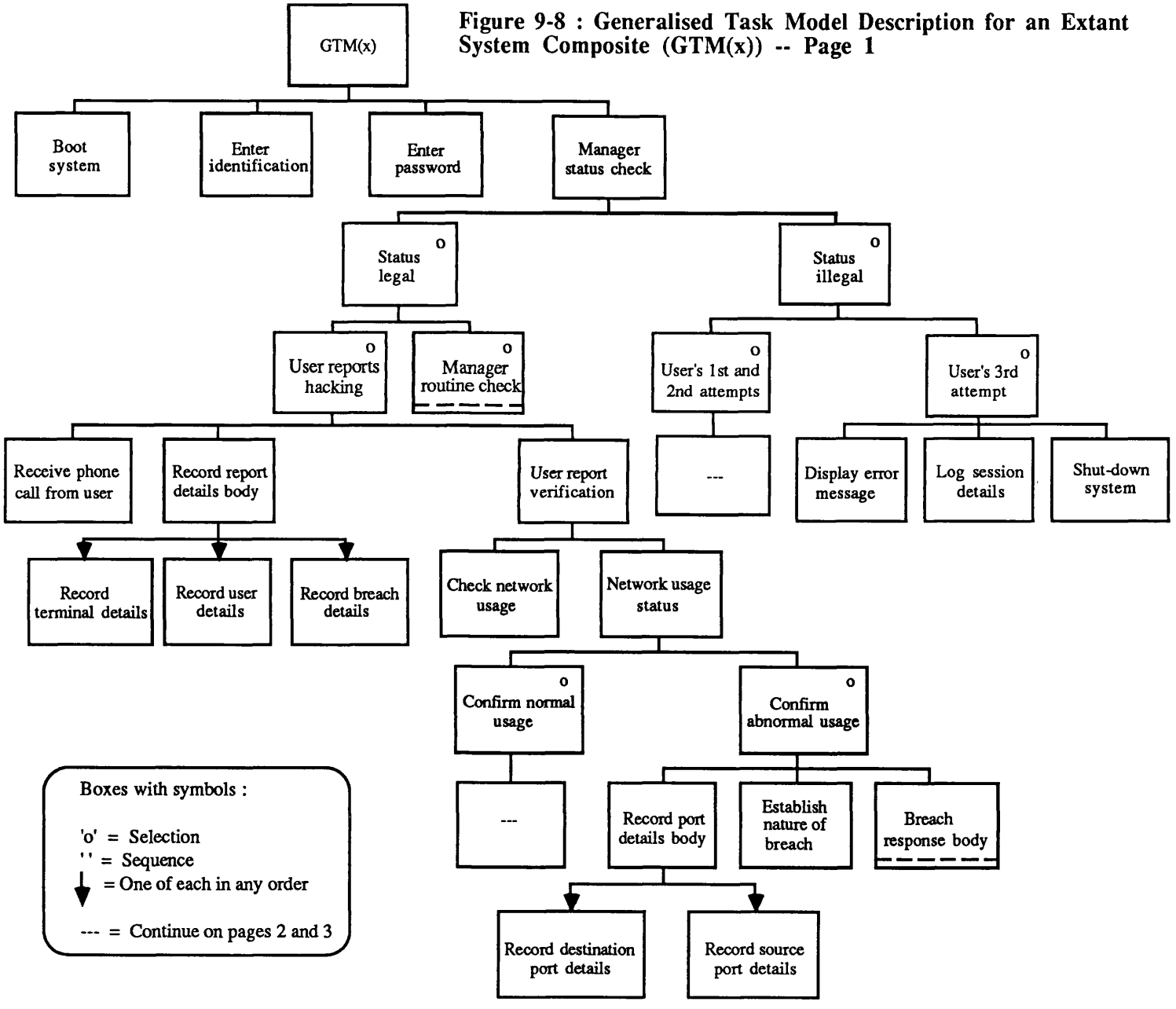


Figure 9-8 : Generalised Task Model Description for an Extant System Composite (GTM(x)) -- Page 1

Figure 9-8 : Generalised Task Model Description for an Extant System Composite (GTM(x)) -- Page 2

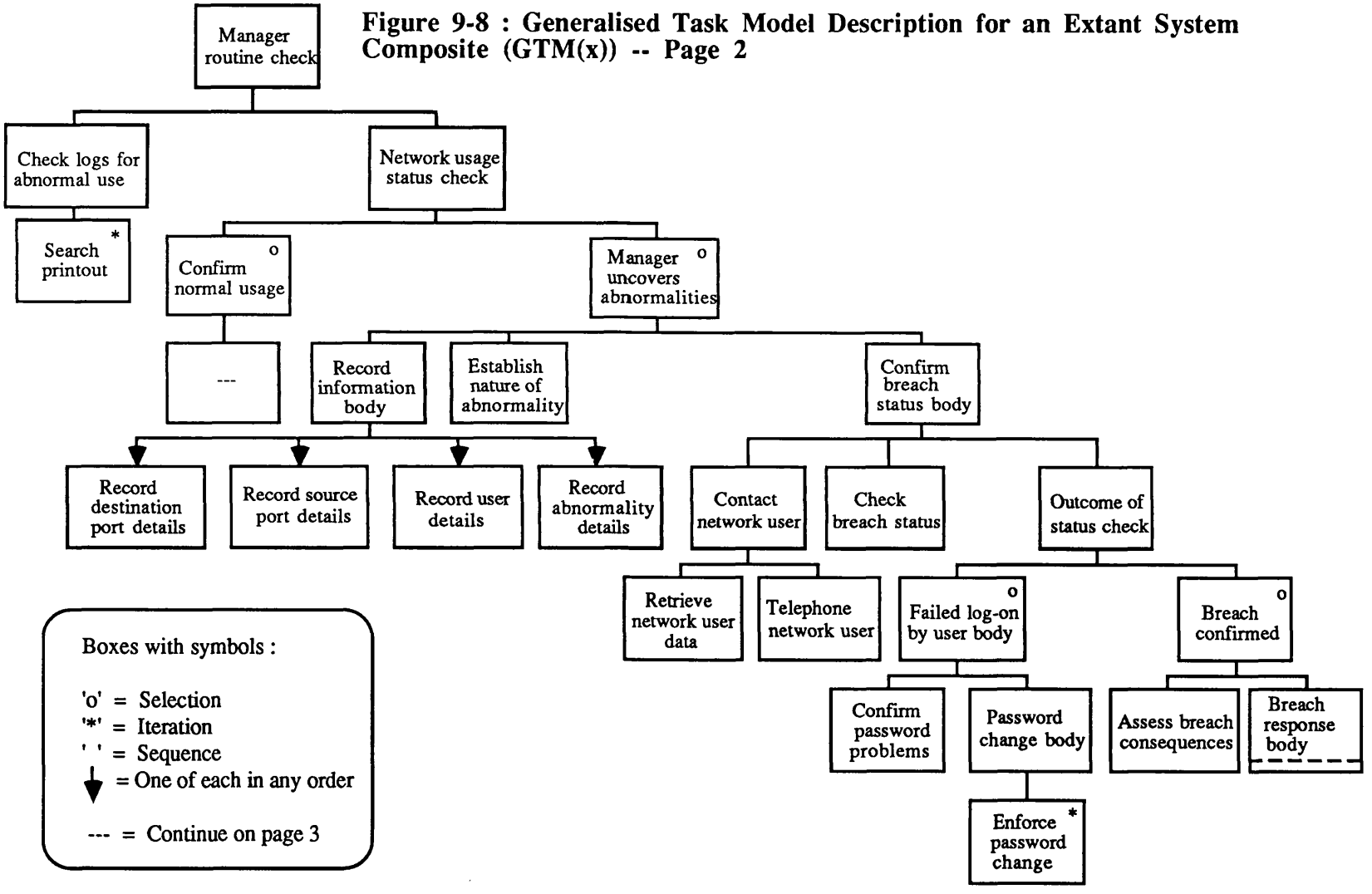
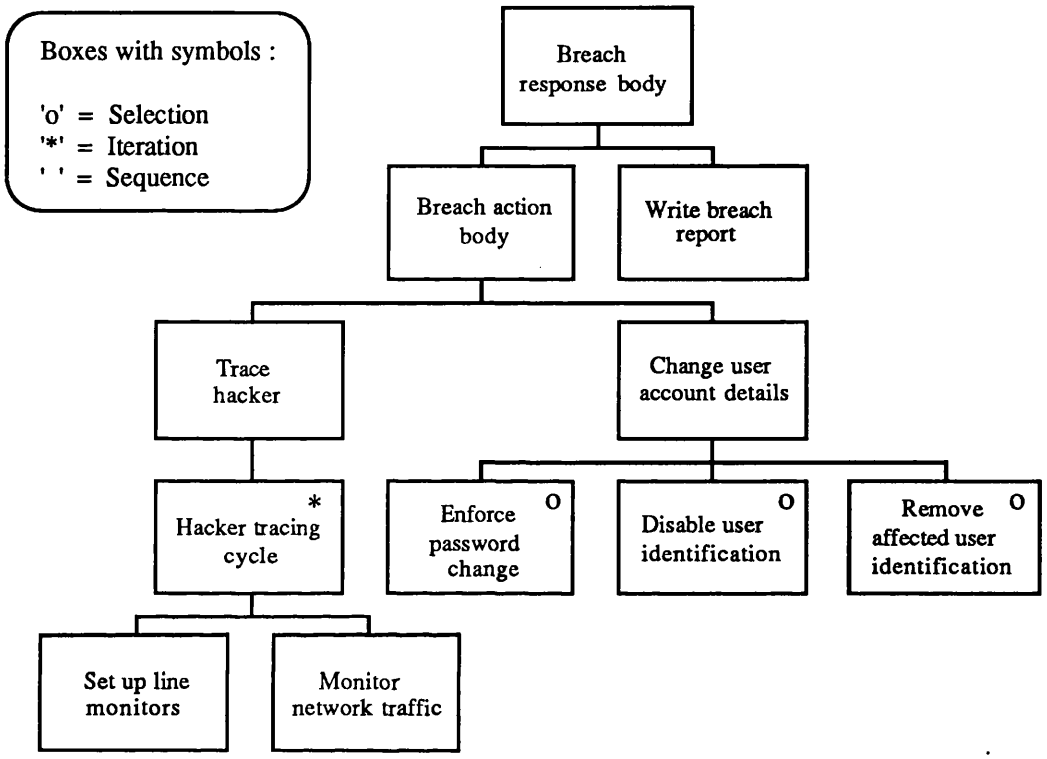


Figure 9-8 : Generalised Task Model Description for an Extant System Composite (GTM(x)) -- Page 3



design criteria for synthesising generalised task model descriptions. Thus, a basis to support subsequent design extensions is established. Such concerns constitute the scope of the Statement of User Needs, Composite Task Model, and System and User Task Model Stages. The design activities and products of these stages are described in the next chapter.

Chapter Ten : The Design Synthesis Phase of the JSD*(HF) Method

"To be still searching what we know not by what we know....."
*Milton, 1644, **Areopagitica**.*

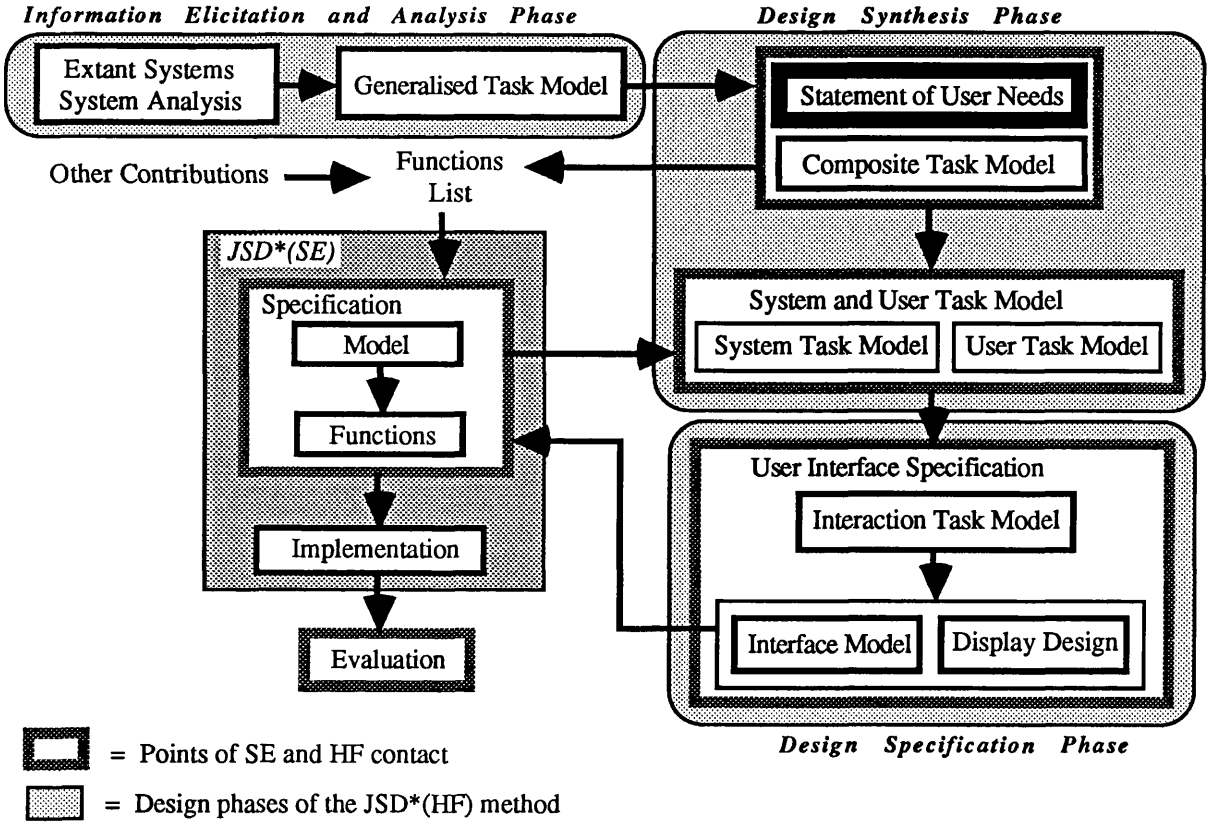
"Leaving the old, both worlds at once they view,
That stand upon the threshold of the new."
Edmund Waller, 1606-1687.

In this chapter, the stages of the Design Synthesis Phase of the method, namely the Statement of User Needs, Composite Task Model and System and User Task Model Stages, will be presented in the order by which design is advanced. The account includes a description of the design products that are derived to support target system specification. Design activities of each of the stages are described using the format outlined in Chapter 8, i.e. each design stage is described as comprising one or more design sub-processes which transform inputs into a number of products. Case-study examples will be used to illustrate these products. In addition, design inter-dependencies between stages of the JSD*(HF) and JSD*(SE) methods are highlighted where appropriate (see Figure overleaf).

10.1. Statement of User Needs (SUN) Stage

Summary

The Statement of User Needs Stage summarizes the conclusions of extant systems analysis and defines user requirements for the target system. Thus, the information collated comprises a mixture of the following : existing user needs and problems; existing design requirements, rationale and constraints; the rationale underlying extant design features to be ported to the target system; performance criteria and domain semantics for the target system; etc. Its location vis-a-vis other stages of the method is highlighted in the Figure (overleaf) by a box outlined in bold.

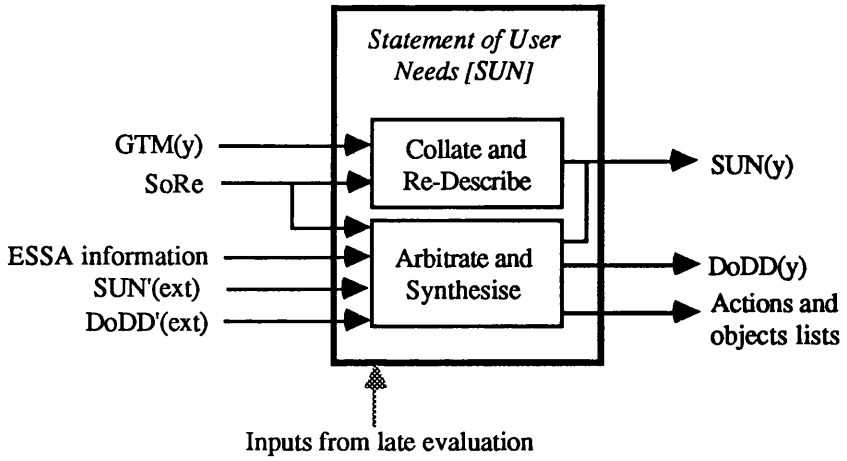


Three design descriptions in respect of the target system are derived at the Statement of User Needs Stage; namely a Statement of User Needs (SUN(y)), a Domain of Design Discourse (DoDD(y)) description and an action and object list. The primary purpose of these products is to constrain later design decisions and extensions, e.g. to ensure an appropriate synthesis of generalised task models at the Composite Task Model Stage.

Detailed Account

The design activities and products of the stage are summarised in Figure 10-1. A detailed account of the products and their derivation follows. As before, design processes shown in the Figure are italicised in the expanded account.

Figure 10-1 : Block Diagram Summary of the Statement of User Needs (SUN) Stage



DoDD = Domain of Design Discourse
ESSA = Extant Systems System Analysis (Stage)
(ext) = JSD(HF) descriptions of an extant systems (EXT)*
GTM = Generalised Task Model
SoRe = Statement of Requirements
(y) = JSD(HF) descriptions of the target system (Y)*

(a) Statement of User Needs for the Target System (SUN(y))

The Statement of User Needs description (SUN(y)) is derived by *collating* and *re-describing* design information extracted from the initial statement of requirements and products of preceding JSD*(HF) stages, e.g. GTM(y). The purpose of the description is to establish a basis for conceptual design extension at later stages of the method. At a minimum, SUN(y) must be detailed enough to support GTM(x) and GTM(y) synthesis at the Composite Task Model Stage. In cases where target system requirements have been addressed in sufficient detail by extant systems analysis, corresponding SUN(ext) descriptions may be incorporated to derive a more detailed SUN(y). For instance, information on specific device-level designs intended for the target system may be included.

As a guide, SUN(y) should address the following :

(i) target system requirements, general design constraints and system performance criteria;
(ii) user problems with the existing system uncovered in earlier HF assessments;
(iii) HF design recommendations and potential solutions to the problems described in (ii) above;
(iv) promising extant design features and the rationale underlying their potential recruitment to the target system.

The scope of the target system is thus characterised initially by a textual description of SUN(y). The description is shared with JSD*(SE) designers since a 'check-point' occurs at this stage of the method (see later account on the Composite Task Model Stage).

The procedures for deriving SUN(y) are summarised below.

Procedures for deriving SUN(y)

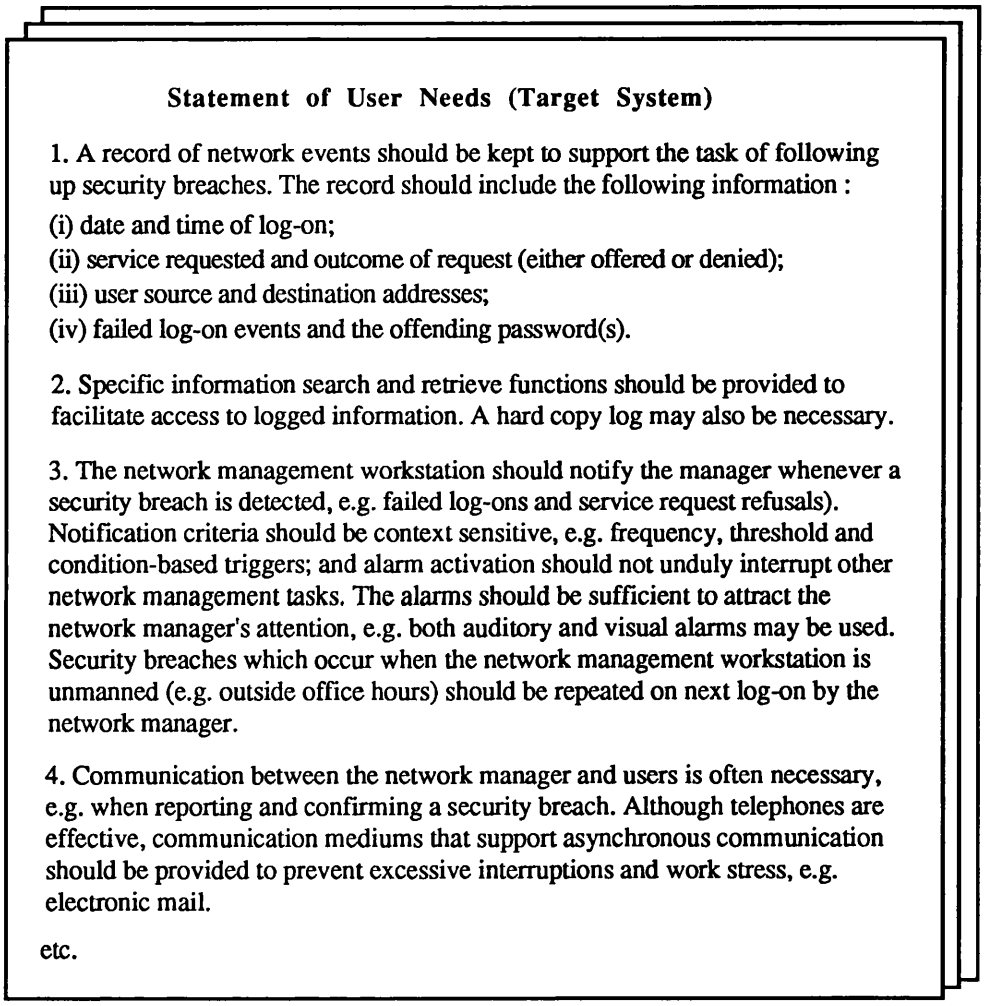
1. *Take as input SUN(ext) descriptions and information on user problems uncovered at the ESSA Stage.*
2. *Interpret user problems with extant systems in the context of the target system.*
3. *Referring to the initial statement of requirements, summarise and extend the statements (e.g. by recruiting descriptions from (1) above) and propose preliminary solutions to the problems highlighted in (1) and (2) above.*
4. *Summarise SUN(y) statements textually.*

Case-Study Illustration of SUN(y)

An extract from SUN(y) for the security management task is shown in Figure 10-2. In this case-study, several statements from the initial statement of requirements and

SUN(ext) descriptions have been incorporated into SUN(y) (compare Figure 10-2 with Figures AD-1 and AD-2 (Annex D)).

Figure 10-2 : Part of SUN(y) for Network Security Management



(b) Domain of Design Discourse Description for the Target System (DoDD(y))

The Domain of Design Discourse description summarises the semantics of the target system by identifying explicit relationships among domain entities (comprising domain objects, major task events and processes). Thus, DoDD(y) establishes a common conceptual scope and vocabulary that support discussions between

designers and end-users. For instance, target system solutions may be proposed and interpreted in terms of common domain entities. In addition, interactions among entities of *real* and *representation* worlds may be compared using such descriptions. Thus, promising metaphors could be assessed for their potential incorporation into a user interface design for the target system (see Frohlich and Luff, 1989; Ragoczei and Hirst, 1990).

A semantic net description of DoDD(y) is constructed by extracting domain information from the initial statement of requirements. Relevant sub-sets of DoDD(ext) descriptions may also be incorporated. To complete the semantic net description, its nodes and relations are expanded textually in an accompanying table.

Procedures for deriving DoDD(y) are summarised below.

Procedures for deriving DoDD(y)

1. Extract domain and task information from the initial statement of target system requirements, GTM(y) and products of extant system analysis (such as DoDD(ext), SUN(ext) and GTM(ext)). The information of interest would comprise the following :
(a) Objects. Object attributes that uniquely define a particular object should be recorded; e.g. 'Network User' attributes = {user id, password};
(b) Task concepts, events and processes, such as 'Hacker Tracing' in network security;
(c) Relations between objects and entities (both composite and taxonomic);
e.g. the 'Network User' entity is general to both 'Hacker' and 'Legal User' entities since it is defined by the network status attribute 'network access' = {true/false}.

2. Collate the information as a semantic net.

Rules of thumb for DoDD(y)

1. DoDD(y) should be sufficiently rich in information to support explanations of system tasks. It should also facilitate scenario construction.

2. DoDD(y) should not include device dependent details. The information described is restricted to the semantics of the target system domain.

3. To finalise a DoDD(y) description, further design iterations may be necessary following agreement (between software engineers and human factors designers) on a Functions List for the target system (see later).

Case-Study Illustration of DoDD(y)

A case-study example of a DoDD(y) description for network security management is shown in Figure 10-3. In this instance, DoDD(y) was synthesised largely from DoDD(ext) descriptions (compare Figure 10-3 with Figures AD-3 and AD-4 (Annex D)). Thus, pertinent sub-sets of extant system descriptions were incorporated with domain information extracted from the initial statement of requirements.

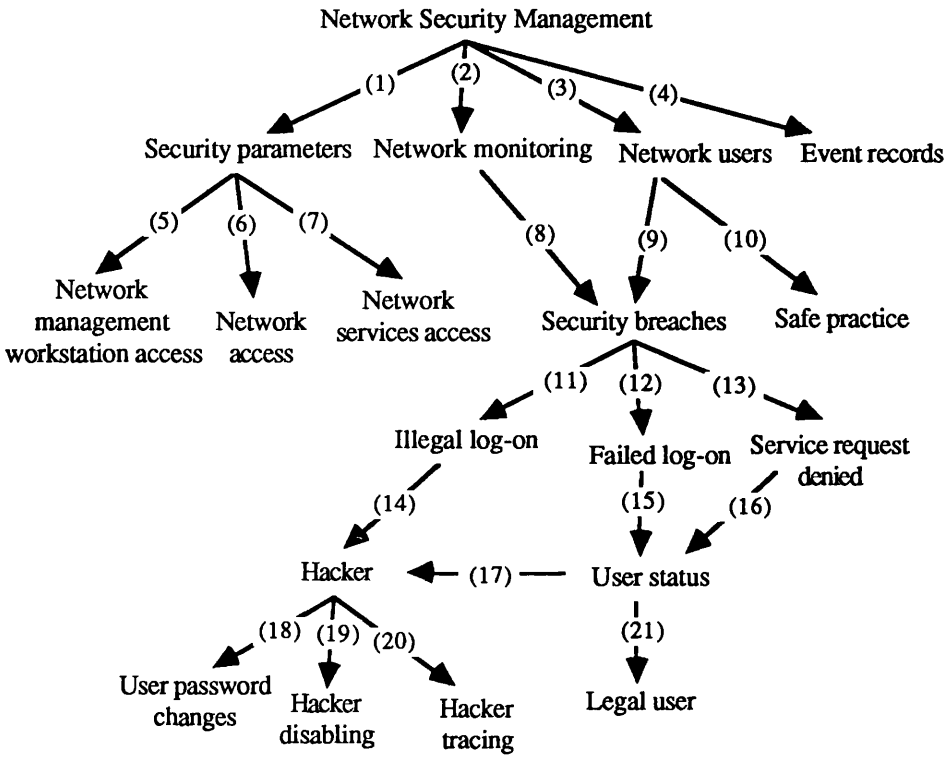
(c) Actions and objects list

The actions and objects list is an *optional* addition to a DoDD(y) description since similar lists are derived at earlier stages of both the JSD*(HF) and JSD*(SE) methods. Since it is an optional product, a case-study example will not be given here (the reader may refer to Lim (1990b) for an illustration). For the present purpose, it suffices to note the following :

(i) the focus of the list should be on the target system domain, i.e. device-specific details should be excluded;
(ii) the list should be collated from the initial statement of requirements and products of extant system analyses.

In conclusion, the objective of the Statement of User Needs Stage is to establish a design basis to support and constrain the specification of an appropriate design solution. For instance, the design basis would include conditions to be satisfied when generalised task models are synthesised to generate a conceptual model of the target system (see later stages of the method).

Figure 10-3 : DoDD(y) for Network Security Management



DoDD(y) Table

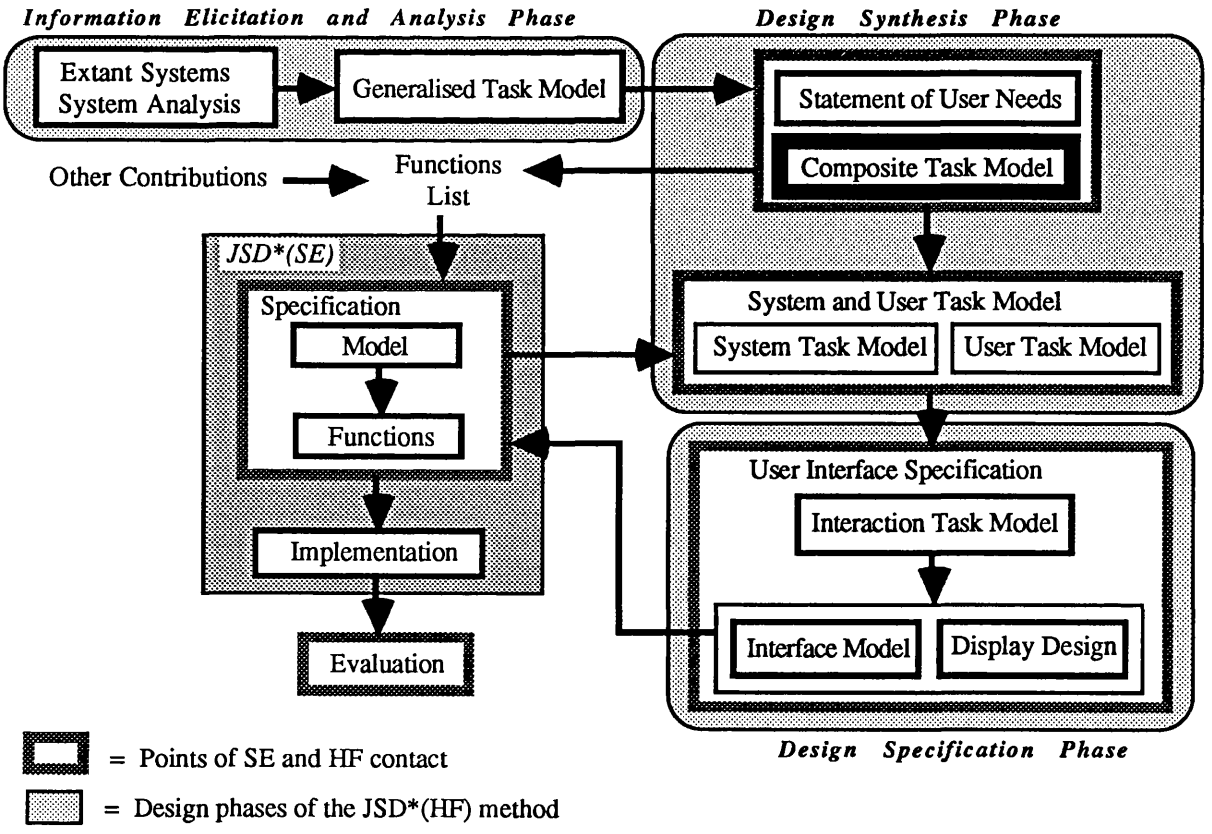
Node	Description	No.	Relation
Network Security Management	The primary concern of network security management is to prevent unauthorised network access.	(1)	To ensure authorised access to the network and its services, the network manager is required to allocate and update security parameters such as user identifications and passwords.
		(2)	Security breaches may be identified by monitoring network access events.
		(3)	Security breaches may be identified by direct liaison with network users. For instance, users may detect and report a security breach on a particular machine to the network manager. Alternatively, the manager may contact the user to ascertain whether suspicious network usage events are attributable to a security breach.
		(4)	Records of network events should be kept. Relevant information includes : source and destination addresses; passwords; user identifications; date and time of log-ons and offs.
Security parameters	etc.	(5)	etc.

10.2. Composite Task Model (CTM) Stage

Summary

The objective of the Composite Task Model Stage is to generate a conceptual model of the target system. Specifically, the generalised task models derived earlier are synthesized to generate a composite task model or a CTM(y) description. On the basis of such a model, specific functions may be allocated to either the human or computer. In other words, components of the composite task model are designated specifically as on-line (i.e. those which are supported by the computer) and off-line (i.e. manual) tasks. On-line sub-tasks may then be decomposed further and demarcated into interactive and computer components as appropriate.

The location of the Composite Task Model Stage vis-a-vis other stages of the method is shown in the Figure below.



Two significant aspects of the Composite Task Model Stage should be highlighted, namely :

(a) design iterations may occur between the Statement of User Needs and Composite Task Model Stages. For instance, such design iterations may be necessitated by modifications to either SUN(y) or CTM(y) following user feedback. Wider design implications may also arise (see (b) below);
(b) the first design inter-dependency occurs between the Composite Task Model and JSD Modelling Stages of the JSD*(HF) and JSD*(SE) methods respectively (see Figure above). At this obligatory 'contact point', software engineers and human factors designers should meet to define the scope of subsequent design extensions. To this end, JSD*(HF) and JSD*(SE) design products generated thus far are discussed. In general, the discussions should consider the following : attributes of objects and actions, user needs and problems, notable task events, task semantics, and the desired support of the user's task. To facilitate the discussions, an event table that highlights major task execution steps may be derived on the basis of the CTM(y) description (see later).[15] The output of these discussions is a list of target system functions (termed the 'Functions List' -- see later). Thus, the scope of target system design is defined and subsequent design extensions by JSD*(HF) and JSD*(SE) designers are hence constrained. In cases where the imposed constraints can not be met satisfactorily, the violations should be notified immediately to other members of the design team. Appropriate design changes may then be introduced to accommodate the violations.

A detailed account of the Composite Task Model Stage follows.

[15] Event table descriptions are predominantly device independent.

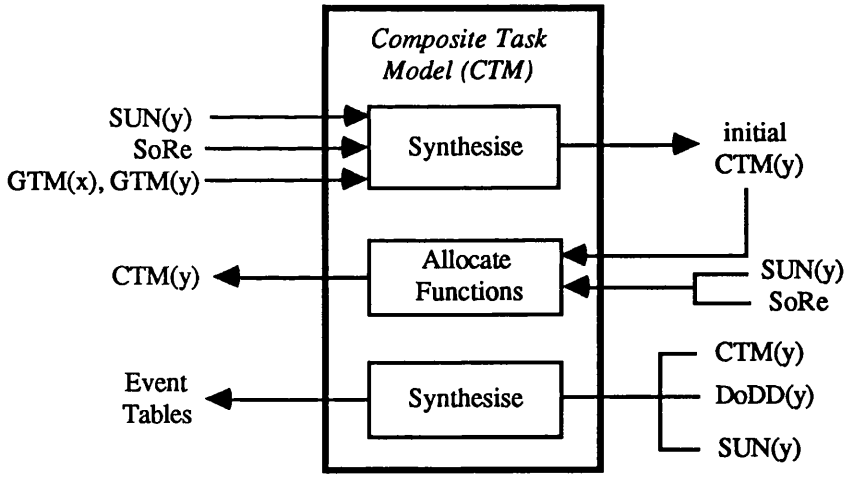
(a) Composite Task Model of the Target System (CTM(y))

The CTM(y) description is generated in two steps (see Figure 10-4). First, the GTM(y) description derived earlier is extended as follows :

(i) novel additions may be proposed to meet new target system requirements;

(ii) pertinent extant system tasks may be incorporated on the basis of the statement of requirements and user needs. Thus, an appropriate sub-set of GTM(x) is *synthesised* with GTM(y)[16] to generate a CTM(y) description.

Figure 10-4 : Block Diagram Summary of the Composite Task Model (CTM) Stage



DoDD = Domain of Design Discourse description

GTM = Generalised Task Model

SoRe = Statement of Requirements *SUN = Statement of User Needs*

(x) = JSD(HF) descriptions of an extant system 'composite' (X)*

(y) = JSD(HF) description of the target system (Y)*

16 The rationale for porting extant design features should be documented so that explicit links may be established with design criteria defined at the Statement of User Needs Stage.

Second, by applying relevant human factors expertise and declarative design guidelines, *functions* are allocated appropriately to the human and computer components of the system. To this end, on-line and off-line tasks are designated by working through each component of the initial CTM(y) description systematically. The scope of the target computer system is thus characterised by on-line components of the CTM(y) description. On the basis of the description, discussions with software engineers may then proceed as per the design inter-dependency requirements of this stage. Following the discussions, the CTM(y) description is updated as appropriate.

Procedures for deriving CTM(y) are detailed below.

Procedures for deriving CTM(y)

1. *On the basis of GTM(y), the initial statement of requirements and SUN(y) identify appropriate GTM(x) components for incorporation. In particular, GTM(x) and GTM(y) descriptions should be compared to identify and remove (or modify) conflicting designs. On a similar basis, consider novel additions to GTM(y) as appropriate. Thus, a structured diagram description of CTM(y) is synthesised at a predominantly device independent level.*
2. *Record the underlying design rationale and decisions in an information table. The structured diagram description of CTM(y) is thus augmented.*
3. *On the basis of earlier HF evaluation of extant system tasks, the initial statement of requirements and SUN(y), allocate functions appropriately between the human and computer. Specifically, components ('leaves') of CTM(y) are designated into on-line or off-line tasks. The designations are differentiated by drawing envelopes around off-line tasks. CTM(y) may have to be decomposed further to ensure an appropriate level of description. To this end, ESSA Stage products may be consulted again as necessary. Continue this process until a satisfactory CTM(y) description is derived.*

Case-Study Illustration of CTM(y)

A case-study example of a CTM(y) description is shown in Figure 10-5 for part of the network security management task. It may be observed from the Figure that an envelope around the structured diagram leaves of CTM(y) indicates off-line tasks. Conversely, on-line tasks correspond to leaves which have no envelope.

For a more specific illustration, consider the occurrence of a failed log-on event (see Figure 10-5 under 'Failed log-on by user body'). On detecting the event, user access is disabled and the network manager is alerted to its occurrence by the computer. The manager is then required to access information logged by the computer to determine the user involved and the cause of the event, e.g. whether the event is due to mis-typing a password or actual attempts at hacking. These tasks, namely access disabling and information gathering are designated as on-line tasks. The manager may also contact the user to verify a particular inference and thus decide whether the disabled user identification should be restored. In the context of the case-study, user contact is unsupported by the computer, i.e. it is an off-line task. Accordingly, the 'Telephone network user' box has a surrounding envelope in Figure 10-5.

To illustrate the extent to which products of extant systems analysis have proved useful for supporting target system design, CTM(y) should be compared with preceding GTM(y) and GTM(x) descriptions (see Figures 10-5, 9-7 and 9-8 respectively). For instance, it may be observed that extant system responses to failed log-on events (as described by GTM(x)) have been ported to the target system (as described by CTM(y)). In particular, both systems involve gathering information on the event and contacting the user to clarify the circumstances of the event. A password change may then be enforced if appropriate. Similarly, specific target system requirements (as described by GTM(y)) have also been incorporated into the CTM(y) description, e.g. automatic security monitoring, and user identification enabling and disabling by the network management workstation.

An information table to support the structured diagram description of CTM(y) is shown in Table 10-2. Thus, lower level details are documented, e.g. the rationale underlying particular characteristics of CTM(y). Such documentation would support stage-wise evaluation, iterative design, and post-implementation design maintenance.

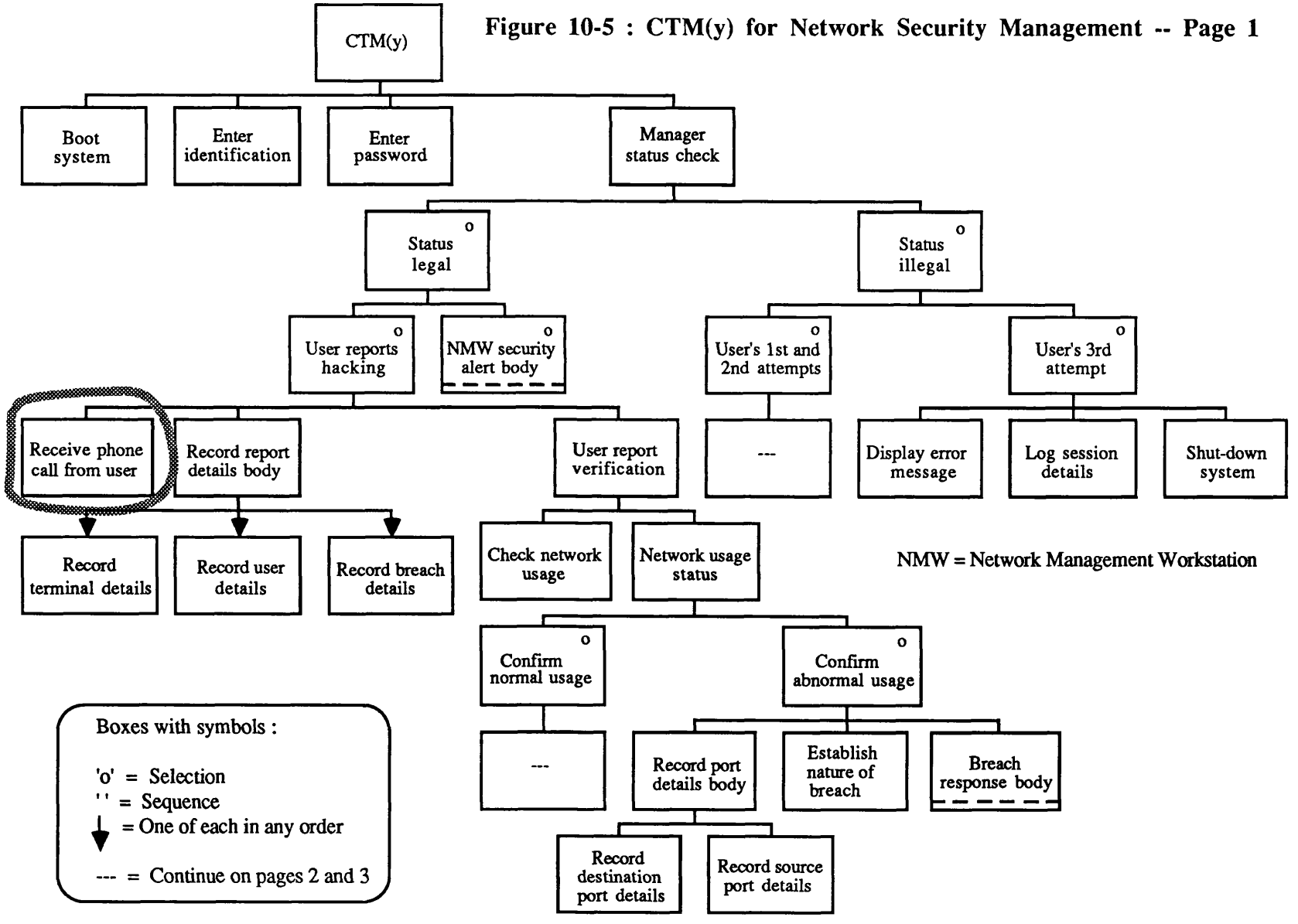


Figure 10-5 : CTM(y) for Network Security Management -- Page 1

Figure 10-5 : CTM(y) for Network Security Management -- Page 2

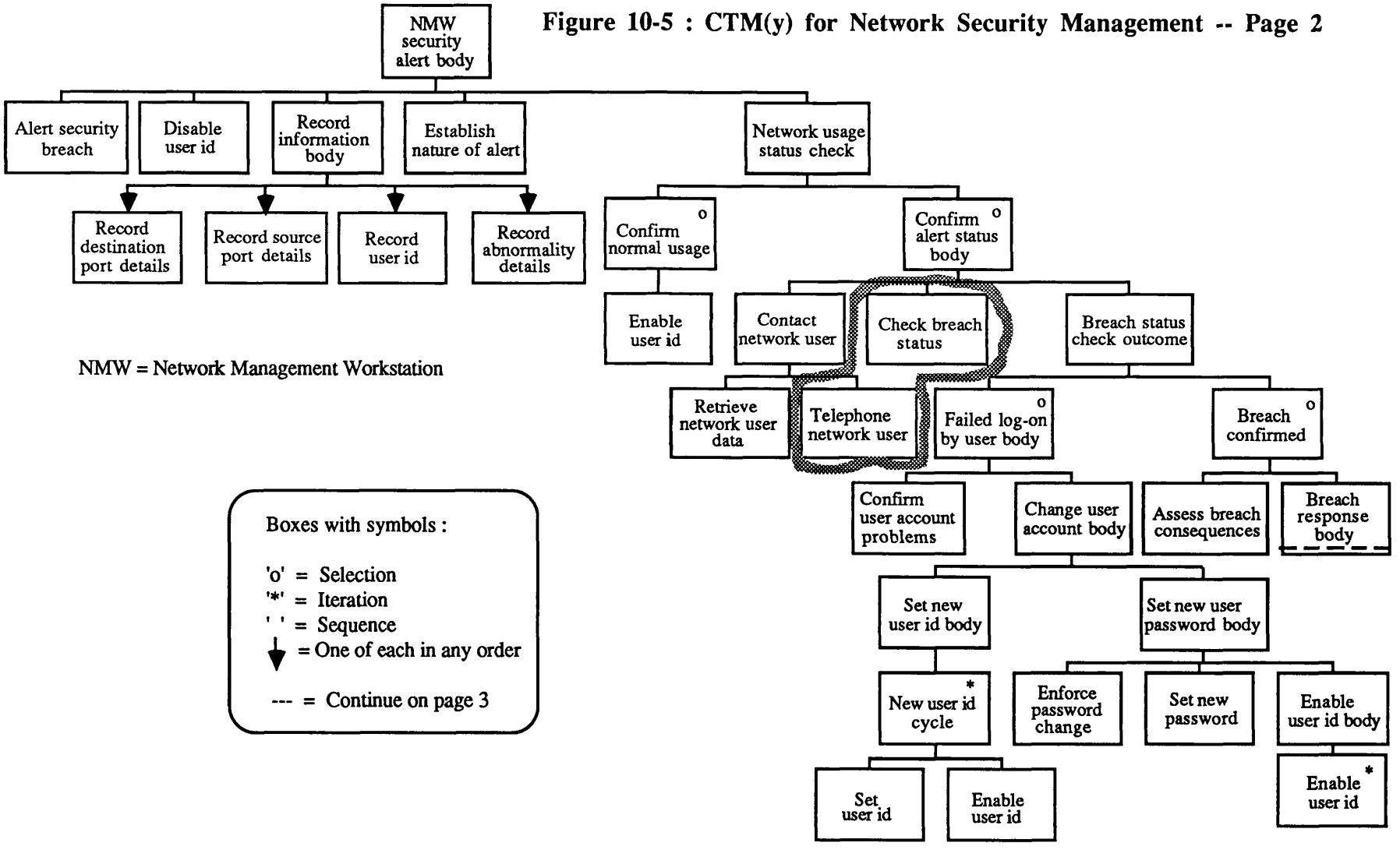
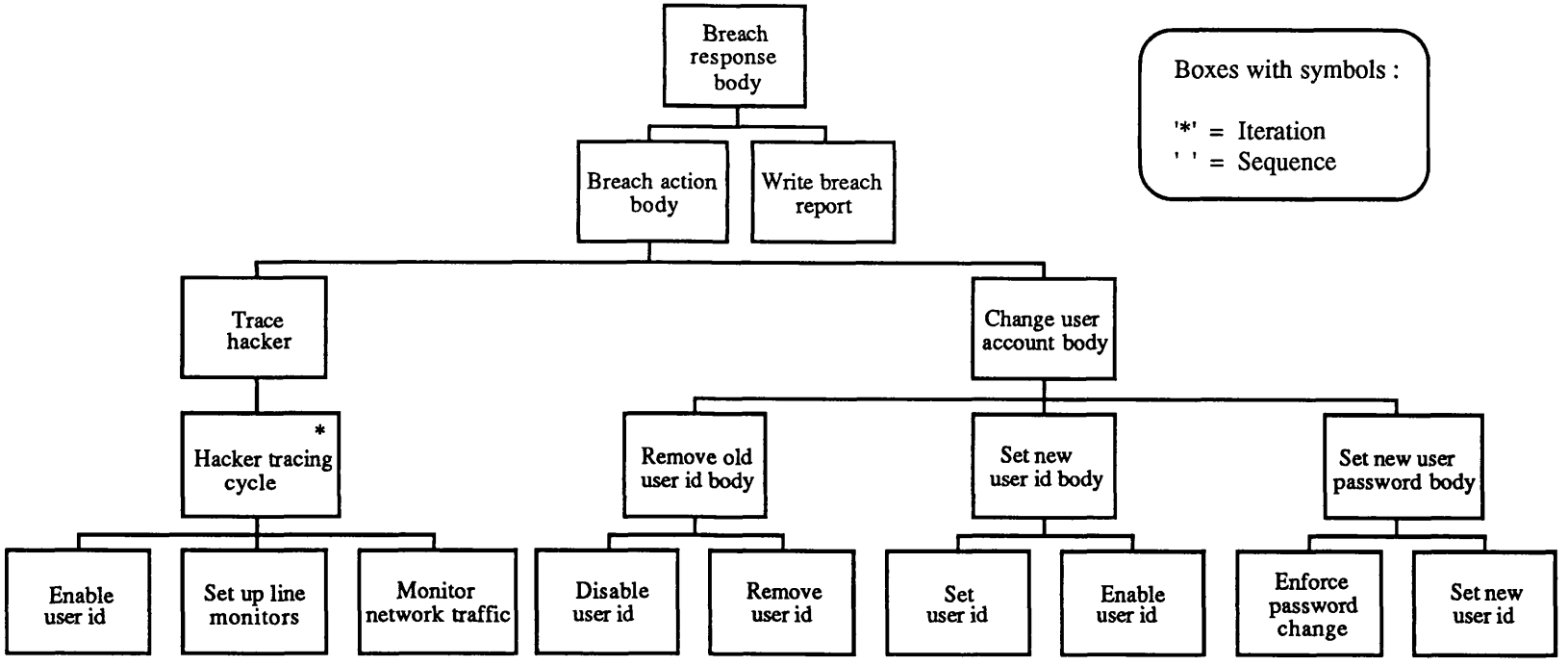


Figure 10-5 : CTM(y) for Network Security Management -- Page 3



CTM(y) Table

Name	Description	Design Comments
Manager status check	The network manager is required to correctly specify an identification number and password to access the network management workstation.	Log-on procedures should offer minimal assistance to potential hackers, e.g. minimal feedback should be provided in between inputs.
Status legal	On workstation access, network security management functions are presented. For instance, the network manager may be alerted by the workstation of the occurrence of a failed log-on event, or the receipt of a user security report.	It is important that contact between the network manager and users is supported adequately since user confirmation of security breach status is frequently necessary.
Retrieve network user data	The network manager must follow up security alerts by consulting the user concerned. Thus, user details such as user identification and contact number should be easily accessible. In this way, the cause of the security alert event may be established.	
Failed log-on by user body	A security alert event will invariably result in automatic disabling of the user identification concerned. The network manager must then contact the user concerned to establish the reason for the event. A decision can then be made as to whether the user identification should be enabled. A password change may also be enforced.	The network manager should be provided with a record of currently disabled user identification numbers and relevant background information. Outstanding actions (e.g. contacting users) can then be reviewed conveniently over time (e.g. weekly).
Enforce password change	The network manager may enforce a password change on a user, e.g. when hacking is suspected but can not be confirmed.	

(b) Event Tables

An event table is an optional product of the Composite Task Model Stage since it does not contribute directly towards design advancement within the JSD*(HF) method. Specifically, its purpose is to facilitate discussions between software engineers and human factors designers. To this end, inter-dependent design information from earlier JSD*(HF) products is collated to generate an event table for the target system. In particular, major task events described by CTM(y), DoDD(y) and SUN(y) descriptions (and extant system contributions to these descriptions) are summarised.

Procedures for deriving an event table are summarised below.

Procedures for deriving event tables

1. Take CTM(y), DoDD(y) and SUN(y) as inputs. If necessary, refer to extant system(s) descriptions and the initial statement of requirements.

2. Identify major events such as sub-task completion and significant real world changes. Note the objects and actions affected by the event.

3. Collate the information in a table.

Case-Study Illustration of an Event Table

A case-study example of part of an event table for network security management is shown in Table 10-3. The table and other JSD*(HF) products (as appropriate) are shared with software engineers during discussions at this inter-dependency point. A more detailed account of inter-dependent design activities follows.

Table 10-3 : Event Table for Network Security Management

Event	Summary	Attributes	Instances
User notifies security breach	Network user indicates security breach has taken place.	-- log-on time and date -- user address -- user identification number -- nature of breach	-- Network manager receives phone call from user.
Consult network event record	A record of network events is examined for evidence of hacker activity, e.g. unusual log-on times.	-- failed log-on -- service request denied -- log-on time	-- Network manager consults network logs.
Trace hacker	Attempts made to locate source address of security breach. May involve tracing across a number of machines.	-- log-on address -- intermediary addresses -- source address	-- Network manager traces hacker via logged data.
Password change	A password change may be enforced on a user following a security breach.	-- user identification number	-- Network manager invalidates user's current password.
Failed log-on	Network user inputs an invalid identification number and/or password. A security breach attempt thus results.	-- user identification number -- number of attempts -- log-on time -- destination and source addresses	

Inter-Dependencies between the JSD Modelling Stage (JSD*(SE) Method) and the Composite Task Model Stage (JSD*(HF) Method)

Generally, a common design scope should be agreed between JSD*(HF) and JSD*(SE) designers at each inter-dependency point, and carried forward through succeeding stages of the JSD* method. Strict adherence to a common scope ensures that subsequent extensions of the target system design are convergent.[17]

To define a common design scope, intersecting JSD*(HF) and JSD*(SE) design concerns and information (i.e. design inter-dependencies) should be identified for discussion. The identification of such intersections is facilitated by the explicit products generated at corresponding stages of the two component methods. Thus, the stage-wise design scope of the methods is examined to identify potential design inter-dependencies. In this way, the first inter-dependency was determined to occur between the JSD Modelling and Composite Task Model Stages of the JSD*(SE) and JSD*(HF) methods respectively. A more detailed discussion of the inference follows.

By definition, the JSD Model (a JSD*(SE) product) describes the purpose and subject matter of the system. Thus, the model focuses on what is to be performed by the system rather than on how work goals may be achieved. Consequently, the design perspectives entailed by JSD modelling and task analysis are different. In particular, user tasks are excluded from the scope of a JSD model. For instance, the JSD Model for a library system is concerned with describing permissible actions on a book rather than a librarian's tasks. Since the actions suffered by a book correspond largely to those initiated by a librarian, potential overlaps between JSD*(SE) and JSD*(HF) methods at this design stage would comprise a set of common domain entities and their actions, e.g. 'shelve' a 'book'. However, in addressing user's tasks and potential user interface design concerns, information about the user and 'representation' world assumed by the current system may also be noted by human factors designers. In contrast, JSD analysts are concerned only

[17] Strict adherence to inter-dependency requirements ensures efficient design management by obviating unnecessary design iterations.

with modelling real world entities. In adopting a minimalist perspective, it may be inferred that the inter-dependent information pool would be dictated by the needs of JSD*(SE) analysts. Thus, JSD*(HF) designers should take account of the information set assumed by JSD*(SE) analysts at this inter-dependency point. However, the converse may not apply. Consequently, at a minimum, the information embedded in the JSD Model should be shared and agreed. The inter-dependent information pool may then be broadened by considering design information requirements of the Functions Stage, and System and User Task Model Stage of the JSD*(SE) and JSD*(HF) methods respectively.

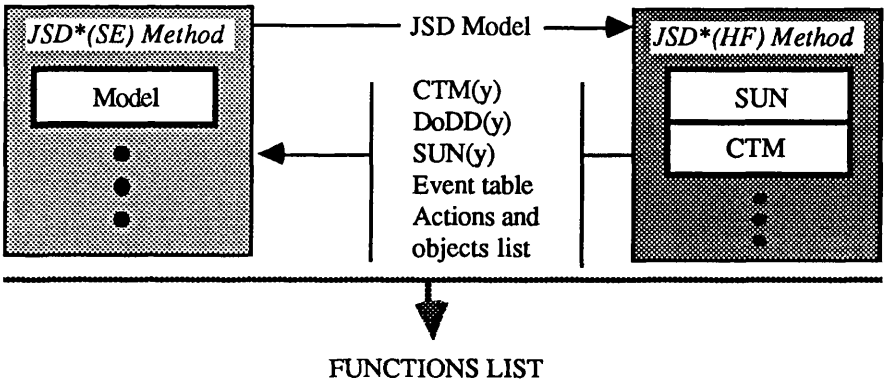
On the basis of such considerations, JSD*(SE) and JSD*(HF) designers should share and agree the following design products at this inter-dependency point :

(i) DoDD(y) : the semantics of the target system are described by this JSD*(HF) product. Thus, it would support the derivation of a JSD model;
(ii) Event table : notable events are listed in the table to characterise the target system scope. The derivation of a JSD model would be supported by such a table;
(iii) SUN(y) : user needs at various levels of description are addressed by this JSD*(HF) product. As such, SUN(y) provides a means of assessing the scope of a particular JSD model with respect to the functions that it should be capable of supporting. In this way, alternative target system boundaries may be investigated;
(iv) Object and action list : the list characterises the domain of the target system and identifies ancillary devices associated with the system. Thus, it supports the derivation of a JSD model;
(v) CTM(y) : a conceptual design of the target system is established by this JSD*(HF) product. In particular, on-line and off-line tasks are demarcated explicitly in a CTM(y) description. Thus, it provides a means of assessing the scope of a particular JSD model with respect to the functions that it should be capable of supporting. In this way, alternative target system boundaries may be investigated;
(vi) JSD model : this JSD*(SE) product provides a software engineer's view of the target system scope. Thus, it complements JSD*(HF) products

described in (i), (iii) and (v) above.

Figure 10-6 is a graphical summary of the above JSD*(SE) and JSD*(HF) products that may be shared at this design inter-dependency point.

Figure 10-6 : Design Products Exchanged between JSD*(SE) and JSD*(HF) Methods at the First Inter-Dependency Point



CTM = Composite Task Model
DoDD = Domain of Design Discourse description
JSD = Jackson System Development
JSD(HF) = Human Factors component of the JSD* method*
JSD(SE) = Software Engineering component of the JSD* method*
SUN = Statement of User Needs
(y) = JSD(HF) descriptions of the target system (Y)*

(c) Functions List

Following discussions and agreement on the scope of the target system, a Functions List is drawn up collaboratively by JSD*(SE) and JSD*(HF) designers. The list summarises the initiating trigger, end result and performance characteristics of task support functions in a tabular format. At this stage, detailed computer functions are usually excluded. On the basis of the Functions List, software engineers and human factors designers may work independently until the next inter-

dependency point (discussed later). It should be emphasised, however, that all designers should be notified of any deviation from the Functions List as soon as they arise.

Case-Study Illustration of a Functions List

A case-study example of part of a Functions List for network security management is shown in Table 10-4.

Table 10-4 : Part of a Functions List for Network Security Management[18]

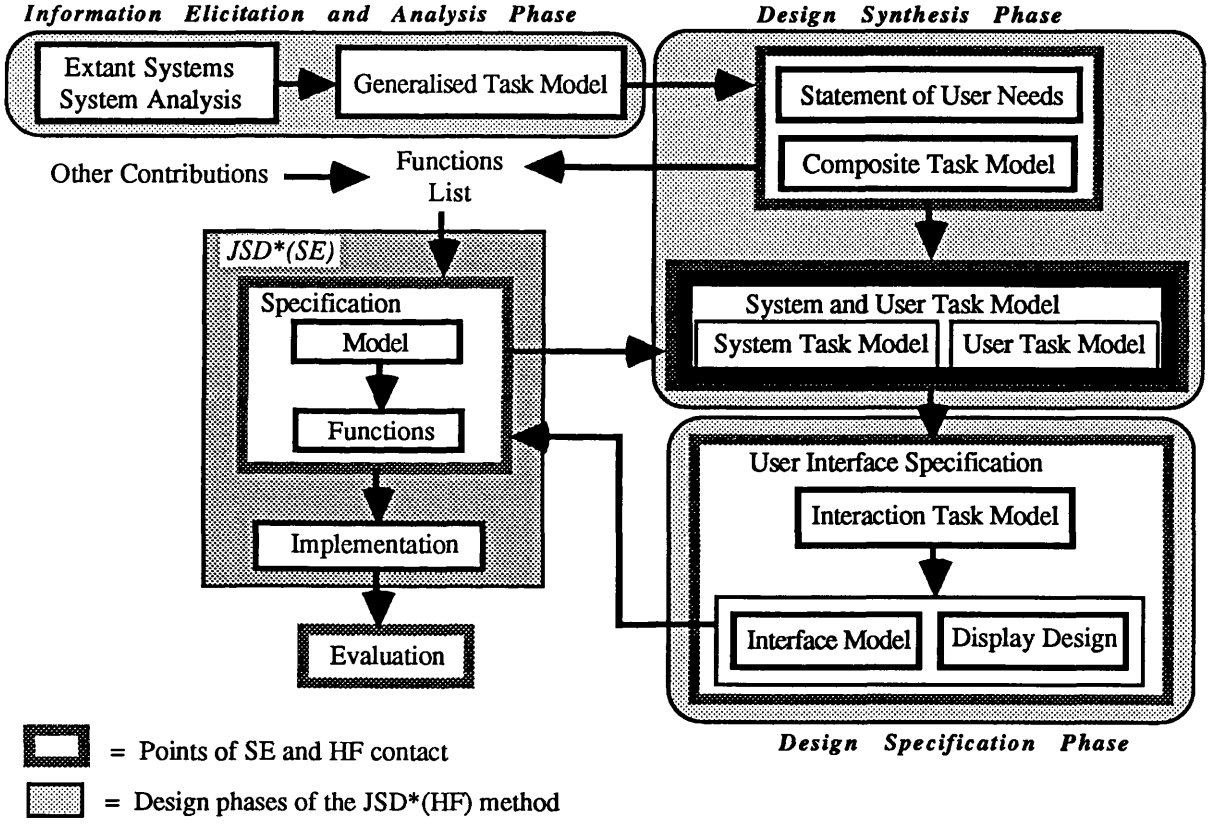
Function	Trigger	End result	Performance
Show network user activity	On network manager's request for information on the activity of a named network user over a specified time period.	Displays information for the named network user over the time period t1-t2 : log-on time, source address, physical location, destination address, log-off time, failed log-on event at time t.	
Alert security breach : failed log-on	Occurrence of a failed log-on event.	Auditory and visual alert to failed log-on event giving time of occurrence, network user identification, access point and physical location.	Within 10 seconds of the failed log-on event.
Record failed log-on events	Occurrence of a failed log-on event.	Record (for previous two months) of failed log-on events giving time of occurrence, user identification, access point and physical location.	
Enforce a password change	On network manager's request to enforce a password change on a specific network user.	On the next log-on, the named network user will be asked to change his/her password. Failure to effect a password change will automatically disable the user identification.	

[18] Table 10-4 is a modified version of a table drawn up by the JSD consultant on the RARDE project (see acknowledgements). The Table limits the extent of RARDE's commitment to the case-study by restricting the scope of JSD*(SE) products to be derived (not described since they are outside the scope of the thesis).

10.3. System and User Task Model (SUTaM) Stage

Summary

Having agreed a common design scope with software engineers, human factors designers may then proceed independently to the System and User Task Model Stage where high level target system functions are specified. The location of the stage vis-a-vis other JSD*(HF) stages is shown in the Figure below.



At the present stage, on-line and off-line tasks of the composite task model (designated previously) are decomposed further to generate system and user task models respectively. An account of these models follows. A system task model is essentially a high level description of the human-computer interaction cycles required to achieve on-line task goals. In contrast, a user task model comprises a summary description of manual tasks. Although, functional design is pursued

primarily by decomposing the human-computer interaction cycles of the system task model, the user task model may also be decomposed further to support workload assessments during job design. It should be noted that during the decomposition of these models, appropriate sub-sets of corresponding extant system descriptions (i.e. STM(x) and UTM(x)) may also be incorporated. To this end, appropriate extant system features are identified by comparing GTM(x) and CTM(y) with respect to the design criteria defined by SUN(y). In particular, relevant STM(x) and UTM(x) sub-sets are determined by identifying specific contributions from GTM(x) to CTM(y).

A further concern at this stage of the JSD*(HF) method involves its design inter-dependencies with the Functions Stage of the JSD*(SE) method. Since functional decomposition is actively pursued by both JSD*(HF) and JSD*(SE) designers at these design stages, the Functions List derived at the previous inter-dependency point would not be specific enough. Thus, further inter-dependencies should be identified to constrain the design extensions to be undertaken presently. In addition, close contact between the designers should be maintained to ensure convergent and efficient design. If close contact can not be achieved, additional design iterations may be necessary when JSD*(SE) and JSD*HF) specifications are integrated. The result is thus inefficient design management.

A more detailed description of the design products and procedures of this stage follows.

Detailed Account

(a) System Task Model of the Target System (STM(y))

STM(y) is derived by *decomposing* on-line task components of CTM(y). During decomposition, due consideration should be given to the design criteria defined at the Statement of User Needs Stage. Thus, the human-computer interaction cycles required for achieving on-line task goals are defined at a high level. These cycles of STM(y) are described by designating its structured diagram leaves into sets of H

(Human) and C (Computer) leaves.

In general, an STM(y) description is derived to support user interface design at the Design Specification Phase of the method. The design support is achieved as follows :

(i) H (Human) leaves of STM(y) may be decomposed further to derive a lower level description of the human-computer interaction required by the on-line task. Such a decomposition is undertaken at the Interaction Task Model Stage where a product, termed ITM(y), is derived (see later). A foundation for specifying required inputs by the computer user is thus established;
(ii) C (Computer) leaves of STM(y) may suggest potential user interface objects for the target system. At the Interface Model Stage, the objects are specified in detail as a set IM(y) descriptions (see later). Thus, a foundation for specifying the following is established :

(1) the behaviour of user interface objects following user input and/or state changes of real and representation world entities;
(2) the appearance of user interface objects and the content of computer messages.

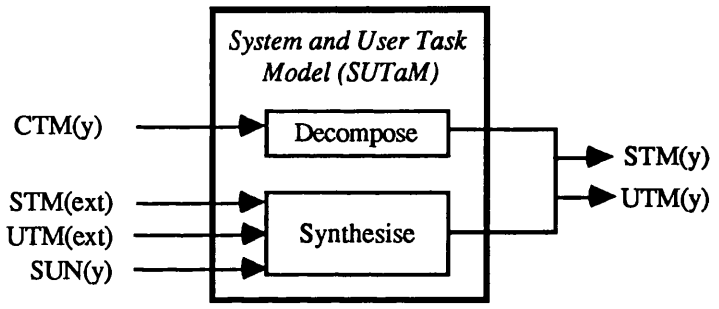
(b) User Task Model of the Target System (UTM(y))

Although off-line tasks do not contribute directly to computer system design (being manual tasks), they should still be considered since their characteristics may have implications for user interface design. For instance, information flows between on-line and off-line tasks may influence the content, format and presentation of computer displays. Consequently, a target user task model (also referred to as UTM(y)) is derived by collating and *decomposing* (if necessary) off-line components of CTM(y). On the basis of the design criteria defined at the Statement of User Needs Stage, appropriate sub-sets of UTM(x) may also be identified for incorporation. Thus, a structured diagram description of UTM(y) is derived. If

necessary, further information may also be documented in a supporting table.

Figure 10-7 summarises the above activities at this stage of the JSD*(HF) method.

Figure 10-7 : Block Diagram Summary of the System and User Task Model (SUTaM) Stage



CTM = Composite Task Model
(ext) = JSD(HF) descriptions of extant systems (EXT)*
SUN = Statement of User Needs
STM = System Task Model
UTM = User Task Model
(y) = JSD(HF) descriptions of the target system (Y)*

The procedures for deriving STM(y) and UTM(y) are described overleaf.

Case-Study Illustration of STM(y)

A case-study example of STM(y) for network security management is shown in Figure 10-8. It can be seen that in deriving STM(y), on-line task components of CTM(y) (see Figure 10-5) have been decomposed and assigned appropriately to human and computer entities of the work system. For instance, the 'Enter user id and password body' sub-node (Figure 10-8 (Page 1), upper-middle part) has been decomposed into network manager (NMgr) inputs and workstation (NMW) prompts. In certain instances, complete separation of the interactive task into H (Human) and C (Computer) leaves may not be necessary (refer to the procedures

for this stage for more information). A case-study example of such an instance is the 'Access user data' leaf which is described as a joint network manager and workstation task (NMgr-NMW) (Figure 10-8 (Page 3), upper left-hand corner).

Procedures for deriving STM(y) and UTM(y) descriptions

1. STM(y) and UTM(y) descriptions are constructed respectively by decomposing on-line and off-line task components of the CTM(y) description. During decomposition, the structure of the CTM(y) description should be maintained as far as is possible to facilitate cross-referencing between STM(y) and UTM(y) descriptions.

2. The STM(y) description should identify Human inputs (H) and Computer outputs (C) required to perform the on-line task. Maintain a 'device independent' description as far as possible.

3. Describe STM(y) and UTM(y) using JSD structured diagrams, and note pertinent design information textually in an accompanying table.*

Rules of thumb for deriving STM(y) and UTM(y) descriptions

1. Significant off-line tasks may be highlighted within an STM(y) description if desired. In such instances, the tasks should be indicated as a structured diagram sub-node and not decomposed further. In this way, cross-referencing between STM(y) and UTM(y) may be supported better. In addition, a more coherent description of the overall task is afforded since the CTM(y) structure is propagated more completely in the STM(y) description.

2. In some cases, STM(y) description may be terminated with a 'H-C' sub-node (rather than distinct H and C leaves). For instance, actions associated with the chosen user interface environment need not be described at the lower level since they are outside the remit of later JSD(HF) design stages. Alternatively, a H-C sub-node may be used where intervening interactive transitions may be assumed as understood. For instance, the sequence comprising 'H: activate function' --> 'C: refresh screen on user input' --> 'C: carry-out function', may be described equally well by 'H-C: carry out function'. The H input required may then be detailed later in the ITM(y) description if necessary. In this way, the STM(y) description would be less cluttered.*

Design Inter-dependencies between the JSD Functions Stage and the System and User Task Model Stage of the JSD*(SE) and JSD*(HF) Methods

A second design inter-dependency occurs between the System and User Task Model Stage and the JSD Functions Stage of the JSD*(HF) and JSD*(SE) methods respectively. This inter-dependency point is extremely important since it addresses the definition and extension of target system functions, e.g. appropriate computer

Figure 10-8 : Part of STM(y) for Network Security Management -- Page 1

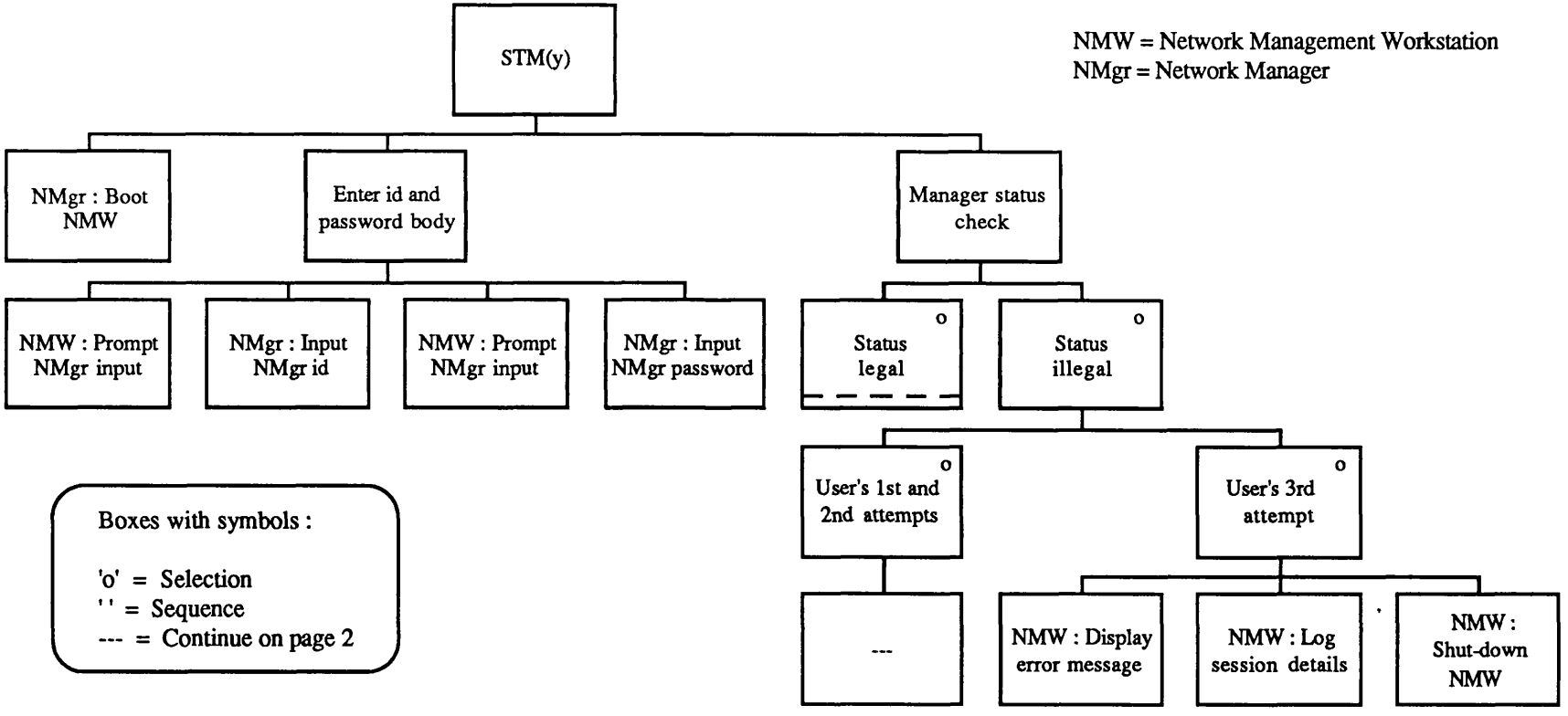


Figure 10-8 : Part of STM(y) for Network Security Management -- Page 2

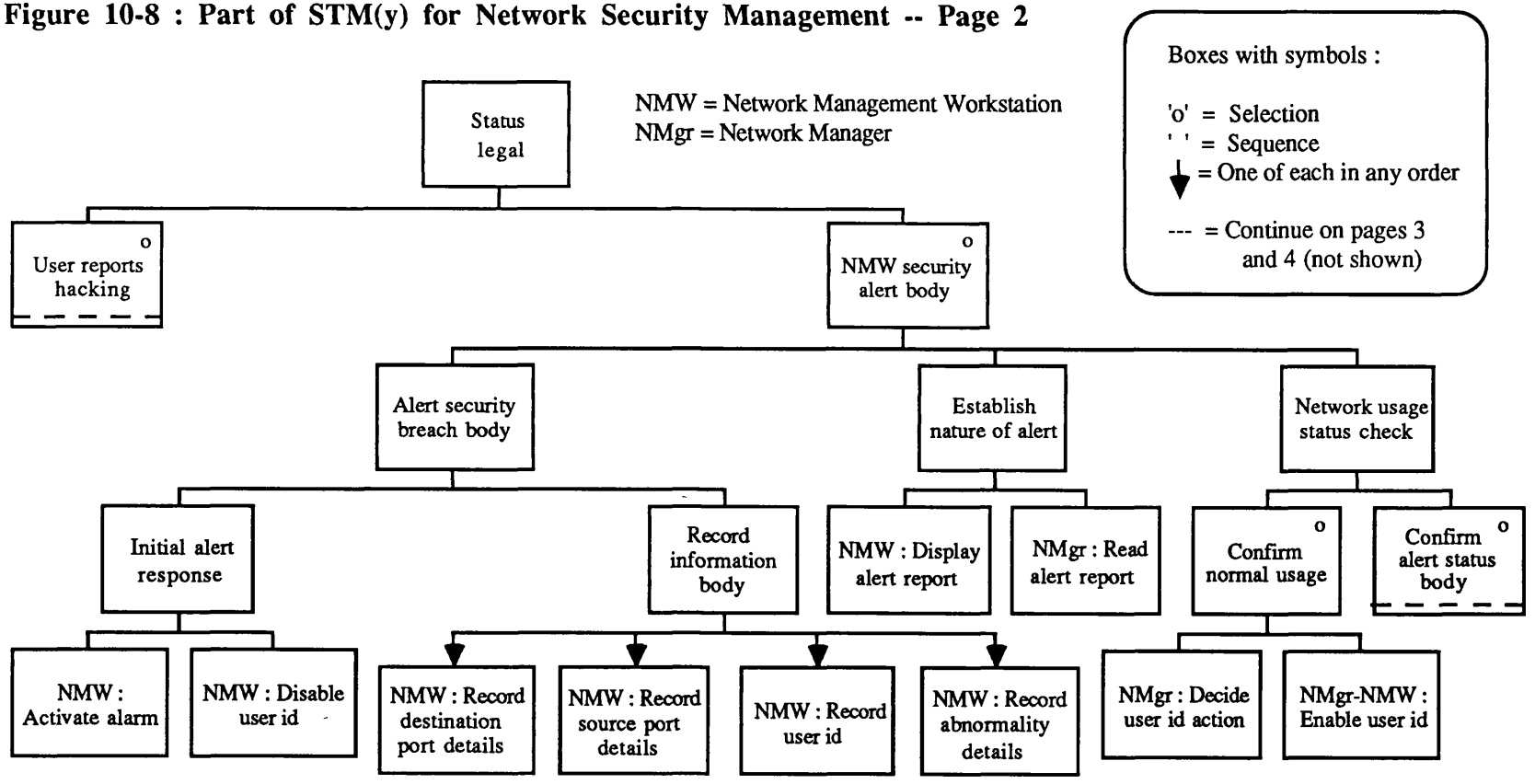
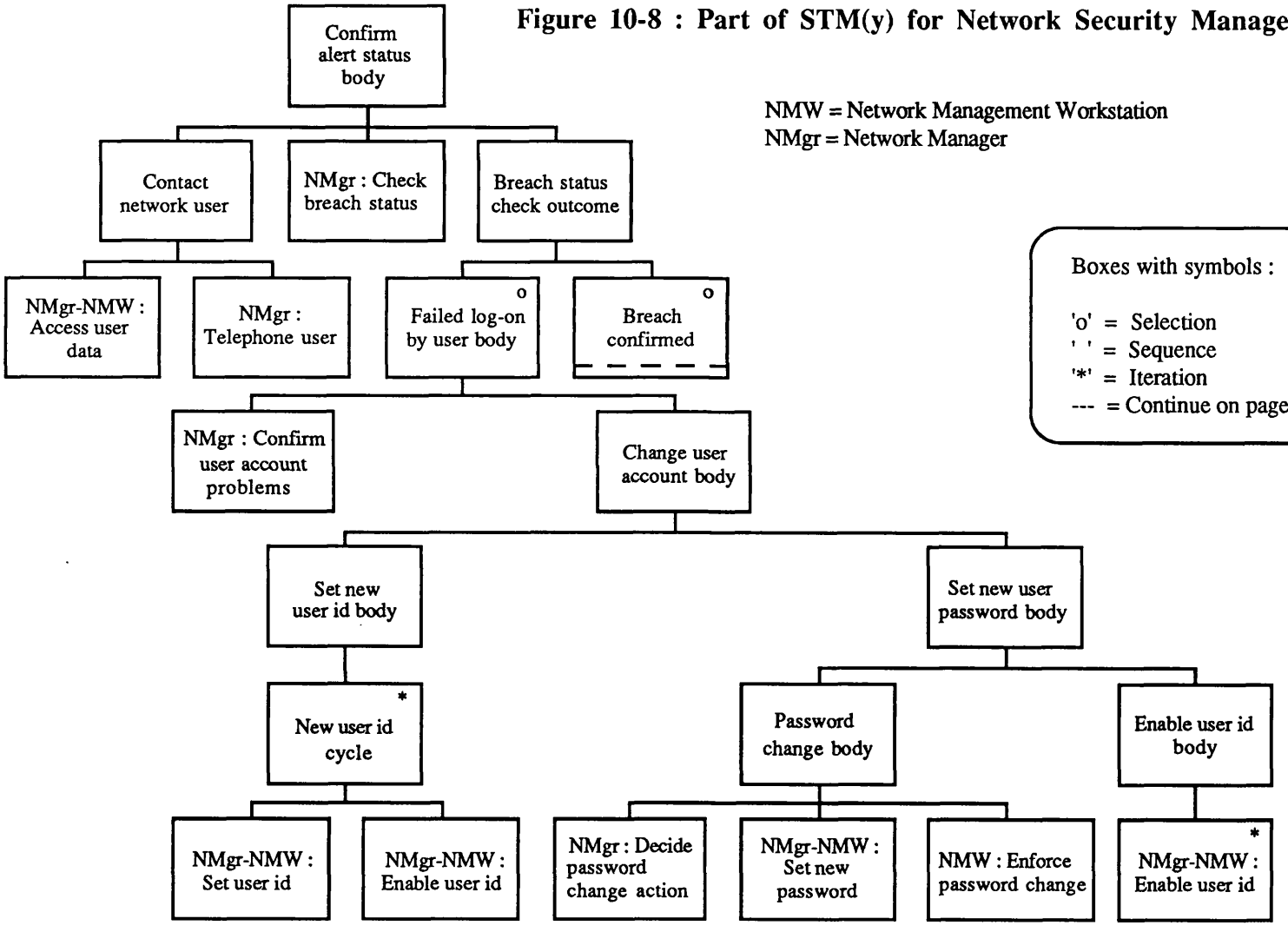


Figure 10-8 : Part of STM(y) for Network Security Management -- Page 3



STM(y) Table

Name	Description	Design Comments
NMW : Prompt NMgr input	The network management workstation prompts the network manager to input a NMgr identification number and password. These inputs are subsequently verified by the network management workstation before NMgr access is enabled.	It is important that input prompts by the network management workstation should offer minimal assistance to those unfamiliar with the log-on procedure.
Alert security breach body	The network management workstation may alert the network manager to security breaches including failed log-ons. The network management workstation can also indicate that a security breach report has been sent by a network user (messaging via the network).	Alerts should be signalled in real time, and should be sufficient to capture the network manager's attention following log-on. Alerts in response to events occurring when the network management workstation is unmanned should be signalled to the network manager on next log-on.
Change user account body	Imposed account changes are enforced automatically by the network management workstation.	

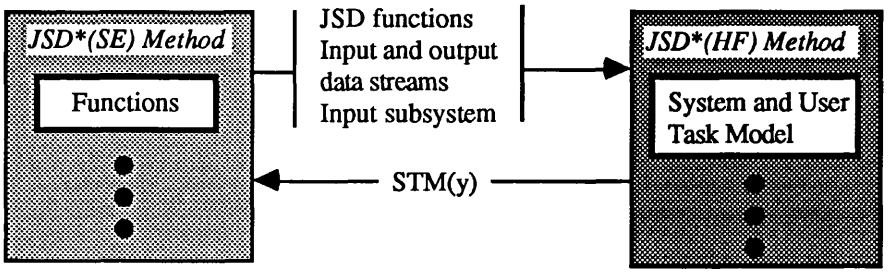
functions are identified to support the user's task; the configurations of JSD functions are decided; etc. Thus, design constraints imposed at the previous inter-dependency point would not be adequate to ensure convergent JSD*(HF) and JSD*(SE) design extensions at this stage. Consequently, further design constraints should be defined.

Figure 10-9 summarises the products that should be shared and agreed between JSD*(HF) and JSD*(SE) designers at this design inter-dependency. The Figure describes the following scenario :

(1) JSD*(HF) designers are expected to contribute a STM(y) description of function sequences required for achieving on-line target system tasks. The sequences are described as a set of high level human-computer interaction cycles;

(2) JSD*(SE) designers are expected to contribute descriptions of the : JSD

Figure 10-9 : Design Products Exchanged between JSD*(SE) and JSD*(HF) Methods at the Second Inter-Dependency Point



JSD = Jackson System Development
JSD(SE) = Software Engineering component of the JSD* method*
JSD(HF) = Human Factors component of the JSD* method*
STM = System Task Model
(y) = JSD(HF) descriptions of the target system (Y)*

function and model processes; JSD input sub-system;[19] and JSD input and output data streams.

These contributions will now be explained further :

(a) JSD functions : since the configuration of JSD functions should support the user's task, pertinent user-related information that has been uncovered by JSD*(HF) designers should be considered by JSD*(SE) analysts at this inter-dependency point. In particular, STM(y), DoDD(y), SUN(y) and CTM(y) descriptions[20] would provide JSD*(SE) analysts with a better view of user needs, problems and task requirements. Thus, a more appropriate set of JSD functions may be specified;

---

[19] Input sub-system specifications contribute to the design of error and feedback messages at later stages of the JSD*(HF) method.

[20] Although it was optional to share the last three JSD*(HF) products at the first inter-dependency point (to pre-empt design information required at the Functions Stage of the JSD*(SE) method), it is obligatory that consensus on these products is reached between JSD*(SE) and JSD*(HF) designers at the present inter-dependency point. Thus, convergence may be ensured between JSD*(HF) and JSD*(SE) design specifications.

(b) System timing : such JSD*(SE) decisions would require a good understanding of user task needs. For instance, decisions concerning the timing and frequency of computer updates of a customer's account should relate to the information required by the user's task, e.g. the cashier would require account updates every minute to enforce withdrawal limits, while the bank clerk would only require daily updates to verify interest computations. To support such decisions, JSD*(HF) design descriptions (namely STM(y), UTM(y), DoDD(y), SUN(y) and CTM(y)) should be consulted for relevant user task information;

(c) JSD data flows : since JSD *input* and *output* information streams implicate exchanges across the user interface, they should be discussed with JSD*(HF) designers (information flows *between* JSD model and function processes are excluded). It is expected that JSD*(HF) designers could then contribute to these JSD*(SE) specifications by sequencing the information streams appropriately with respect to the user's interactive task;

(d) JSD input sub-system : JSD*(SE) specifications of the input sub-system include error categories (e.g. simple errors and false inputs) and context filters. The specifications should be discussed with JSD*(HF) designers since they intersect later human factors specifications of error and feedback messages.

In addition to establishing a consensus on the above JSD* products, close contact should be maintained between JSD*(HF) and JSD*(SE) designers at this inter-dependency point. Thus, new design extensions should be communicated between designers as soon as they are developed sufficiently.

The above account completes a stage-wise review of the Design Synthesis Phase of the JSD*(HF) method.

At this juncture, a conceptual target system design would have been specified sufficiently for user interface design to proceed. The latter is addressed at the Design Specification Phase in three stages, namely the Interaction Task Model, Interface Model and Display Design Stages. A account of these stages follows.

Chapter Eleven : The Design Specification Phase of the JSD*(HF) Method

"The end of our foundation is the knowledge of causes, and the secret motions of things; and the enlarging of the bounds of human Empire, to the effecting of all things possible."

*Francis Bacon,, 1627, **New Atlantis.***

Following the conceptual design of the target system, user interface specification commences in the Design Specification Phase of the JSD*(HF) method. Presently, the three stages that comprise this phase, namely the Interaction Task Model, Interface Model and Display Design Stages, are presented in the sequence performed during design (i.e. in the above-mentioned order). As before, design activities and products of the stages are summarised using a block diagram. Case-study examples are also provided where appropriate.

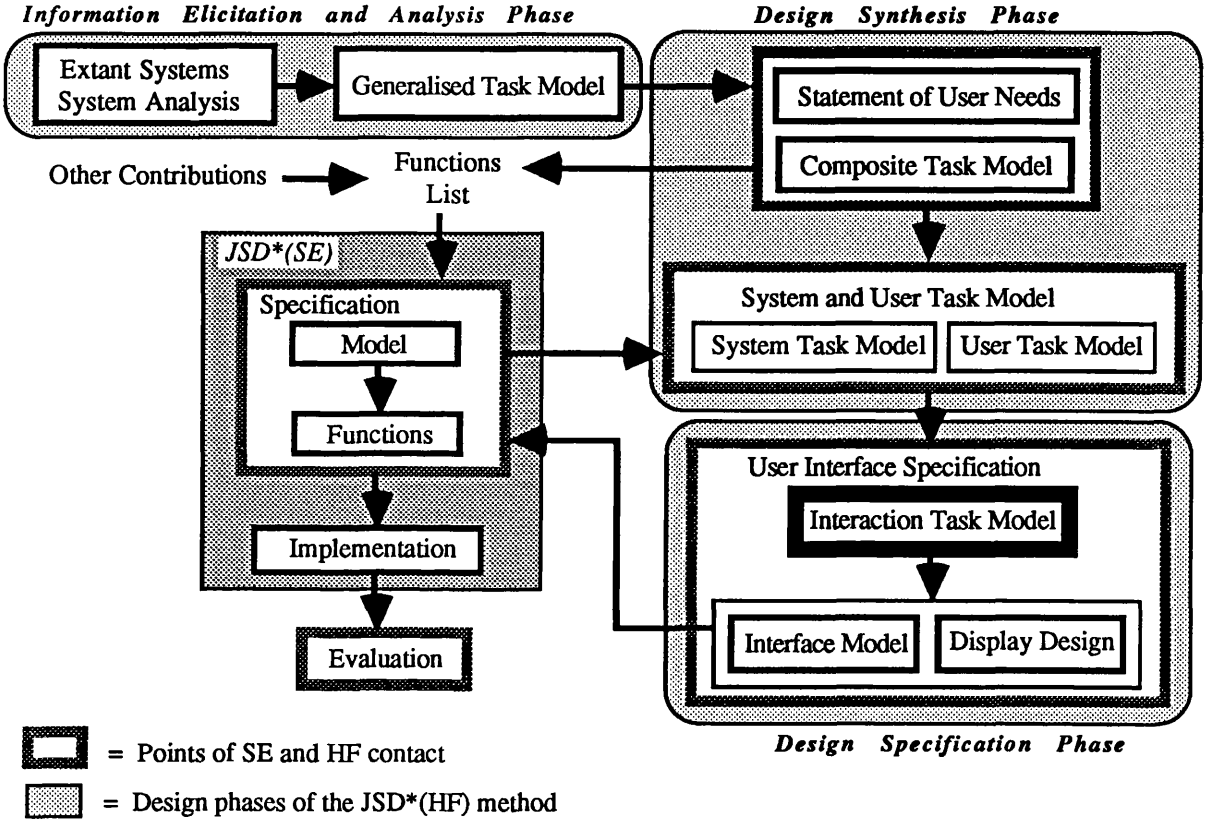
11.1. Interaction Task Model (ITM) Stage

Summary

Having defined the on-line task conceptually in terms of human-computer interaction cycles of an STM(y) description, the cycles may be decomposed further at the Interaction Task Model Stage. A product, termed an interaction task model (or ITM(y)), is derived. The location of this stage vis-a-vis other stages of the JSD*(HF) method is shown in the Figure overleaf.

ITM(y) is essentially a description of the device level interactions required to achieve user task goals using the target computer system. It is described in terms of object and action primitives of the chosen user interface environment (if any) and basic keystrokes of the designated hardware. To support subsequent specification of error recovery schemes, feedback messages and screen displays, low level actions of ITM(y) are also grouped into coherent interaction 'units'. In this respect,

it is essential that an appropriate level of ITM(y) description is derived. To this end, design iterations with later JSD*(HF) stages may be expected.

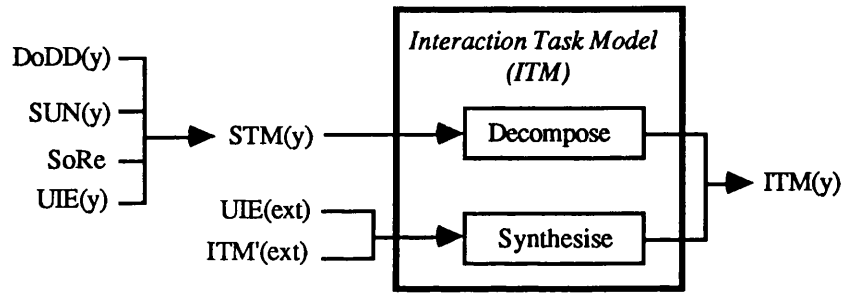


Detailed Account

Since the Interaction Task Model Stage follows the System and User Task Model Stage, STM(y) comprises its primary input. Specifically, H leaves of STM(y) (i.e. on-line user actions) are *decomposed* further to derive a device dependent description comprising object and action primitives of the chosen user-interface environment (if any) and basic keystrokes of the designated hardware. To ensure the derivation of a consistent ITM(y) description, earlier JSD*(HF) products, namely SUN(y) and DoDD(y), should also be considered during STM(y) decomposition. In addition, relevant aspects of the current user interface environment (if any) and an appropriate sub-set of ITM'(ext) (extant system description derived at the Extant Systems System Analysis Stage) may be *synthesised* iteratively with an initial ITM(y) description (see Figure 11-1). Thus,

an ITM(y) description sufficient to support later JSD*(HF) design stages (namely the Interface Model and Display Design Stages) is derived. It is essential that the final ITM(y) description should be at a level understood generally by design team members.

Figure 11-1 : Block Diagram Summary of the Interaction Task Model (ITM) Stage



DoDD = Domain of Design Discourse
(ext) = JSD(HF) description of extant systems (EXT)*
SoRe = Enhanced Statement of Requirements
SUN = Statement of User Needs
(y) = JSD(HF) description of the target system (Y)*
STM = System Task Model
UIE = User Interface Environment

On deriving a satisfactory description of ITM(y), screen 'boundaries' may be designated at appropriate intervals between groups of structured diagram leaves (see case-study example later). In other words, appropriate start- and end-points of interactive task units are demarcated on the ITM(y) description. For each designated 'boundary', unique numbers (e.g. 'bubble' S1 in Figure 11-2) are assigned to support cross-referencing between ITM(y) and products of the Interface Model and Display Design Stages. Explicit links are thus established between the error-free task description of ITM(y), and static and dynamic descriptions of screen presentation specified at these stages. In this way, the actuation of particular screens (of defined composition and layout) is set appropriately against the user task context (see later). Thus, the presentation context for computer support functions, and error and help messages is defined.

Procedures for deriving ITM(y) are described below.

Procedures for deriving ITM(y)

1. Select H and H-C leaves of the STM(y) description for further decomposition.

2. Decompose each H leaf (or H-C leaf) to a level that is easily understood by design team members. To ensure a consistent design description, the following should be considered during decomposition :

(a) characteristics of the extant and chosen target user interface environment (if any), e.g. when naming newly specified actions of ITM(y);

(b) extant interaction task model (ITM(ext)) as appropriate;

(c) DoDD(y), SUN(y), and enhanced statement of requirements for the target system.

3. Note specific design features of ITM(y) to be considered later when products of the Interface Model and Display Design Stages are derived, i.e. IM(y) and DD(y).

4. Describe ITM(y) using JSD structured diagram notation and note additional information in an accompanying table.*

5. Re-work CTM(y), STM(y) and UTM(y) as necessary. To this end, iterations with preceding design stages may be necessary. In some cases, wider changes may involve modifying the original system task allocation. Thus, close contact with JSD(SE) analysts should be maintained, i.e. changes should be communicated between designers as they arise so that HF and SE design converge efficiently.*

6. On deriving a satisfactory description of ITM(y), work through sections of the structured diagram systematically as per the constructs of the notation. Specifically, screen boundaries are demarcated on the ITM(y) description by identifying coherent groups of interactive task units in the STM(y) description. Assign alpha-numeric identifiers to each boundary so that inter-linkages may be established later between ITM(y) and products of the Design Display Stage, e.g. Pictorial Screen Layouts (refer to the procedures of the Design Display Stage for an account of the indexing scheme used for this purpose). Continue the process for the entire ITM(y) description.

Rules of thumb for deriving ITM(y)

1. It is vital to derive a satisfactory description of ITM(y) before the Interface Model and Display Design Stages are undertaken.

2. Several versions of ITM(y) may be necessary before a satisfactory description is derived. In other words, ITM(y) derivation may involve decomposing STM(y) in two or more steps. For instance, ITM(y) may be described initially in terms of input primitives of the chosen user interface environment (if any). Following the derivation of a set of IM(y) descriptions and Pictorial Screen Layouts at later stages of the method, the initial version of ITM(y) may then be decomposed further to detail inputs associated with bespoke user interface design features. Iterations with later design stages may also be necessary.

3. Sub-nodes of STM(y) at two levels (or more) from the bottom of the structured diagram description are likely to remain unchanged, i.e. in most instances, they would be carried over to the ITM(y) description.

4. To ensure unique sub-node labels in a structured diagram description, it may be necessary to modify sub-node names that have been carried forward from the STM(y) to the ITM(y) description (see (3) above). The original and new names should be semantically similar so that relationships between STM(y) and ITM(y) descriptions remain clearly identifiable.

In conclusion, the objectives of the Interaction Task Model Stage are three-fold :

(a) to describe the device level interactions that a user is expected to perform using the target computer system;
(b) to establish a foundation for advancing design at later stages of the method; and
(c) to specify a reference framework that inter-links design products derived at the Design Specification Phase of the method.

Case-Study Illustration of ITM(y)

A case-study example of ITM(y) for network security management is shown in Figure 11-2. This description of ITM(y) was derived following several iterations with later stages of the method (refer to the procedures of the Interface Model and Display Design Stages for further information). Thus, user inputs required for accomplishing on-line task goals were specified at a lower level of description. Note that a supporting information table was not derived on this occasion since the structured diagram provided a sufficiently clear description of ITM(y).

It was indicated earlier that ITM(y) is derived by decomposing H leaves of STM(y) (i.e. human actions) in terms of input primitives of the chosen user interface environment. For the present case-study, HyperCard™ was used to simulate the implementation environment.[21] Thus, the 'Access user data' action of STM(y) (Figure 10-8 -- Page 3) was decomposed to derive an ITM(y) description that includes HyperCard™ primitives such as 'button' objects and 'button click' actions (see Figure 11-2 -- Page 1).

Figure 11-2 illustrates another characteristic of an ITM(y) description, namely the demarcation of screen boundaries and numbering scheme, e.g. S1, S2, etc. Presently, a case-study example of how such screen demarcations may be linked to pictorial screen layouts is described for Screens 3A and 3B (corresponding to

[21] HyperCard™ was chosen because it is a tool commonly used for prototyping WIMP-type user interface designs.

Figure 11-2 : Part of ITM(y) for Network Security Management -- Page 1

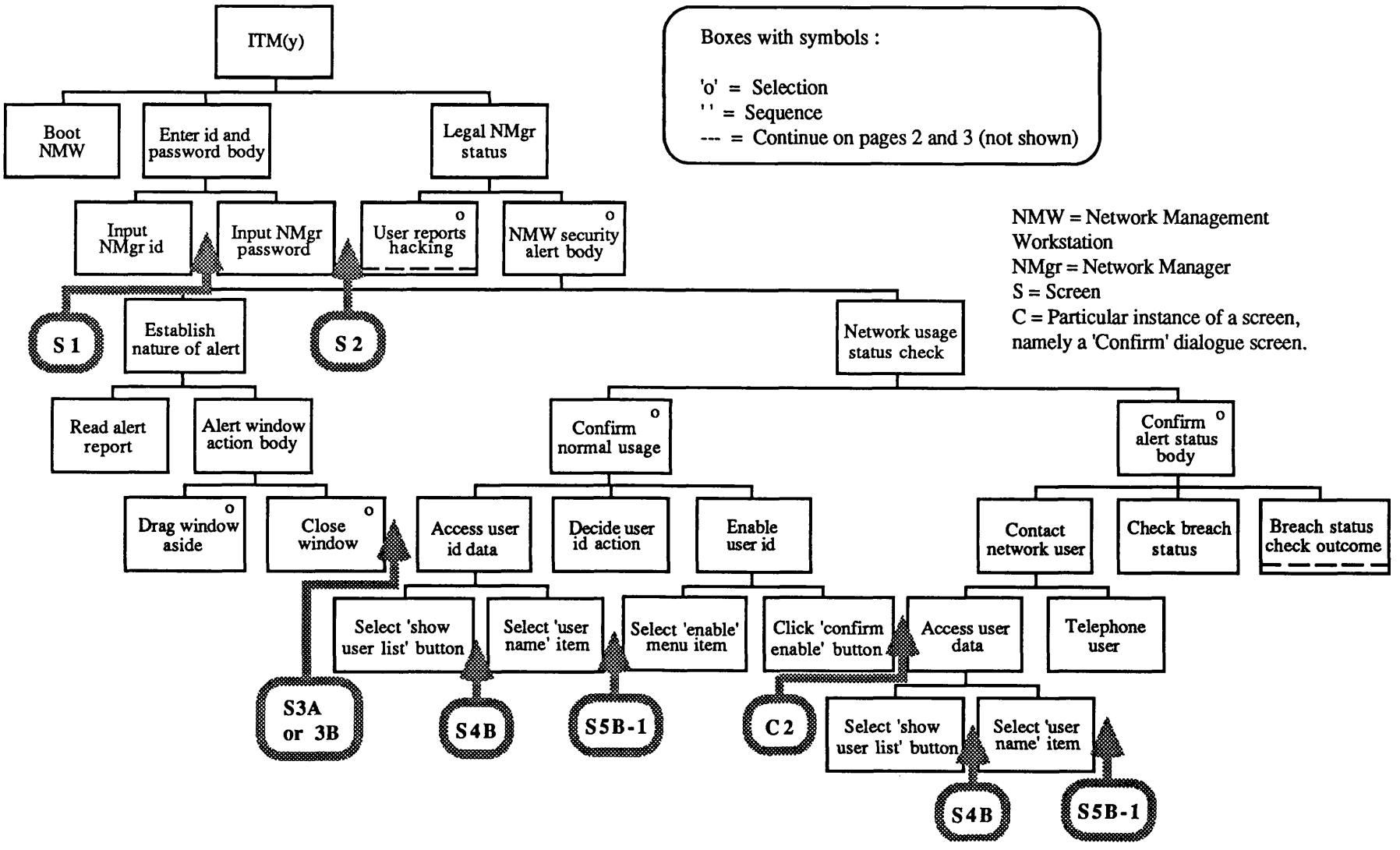
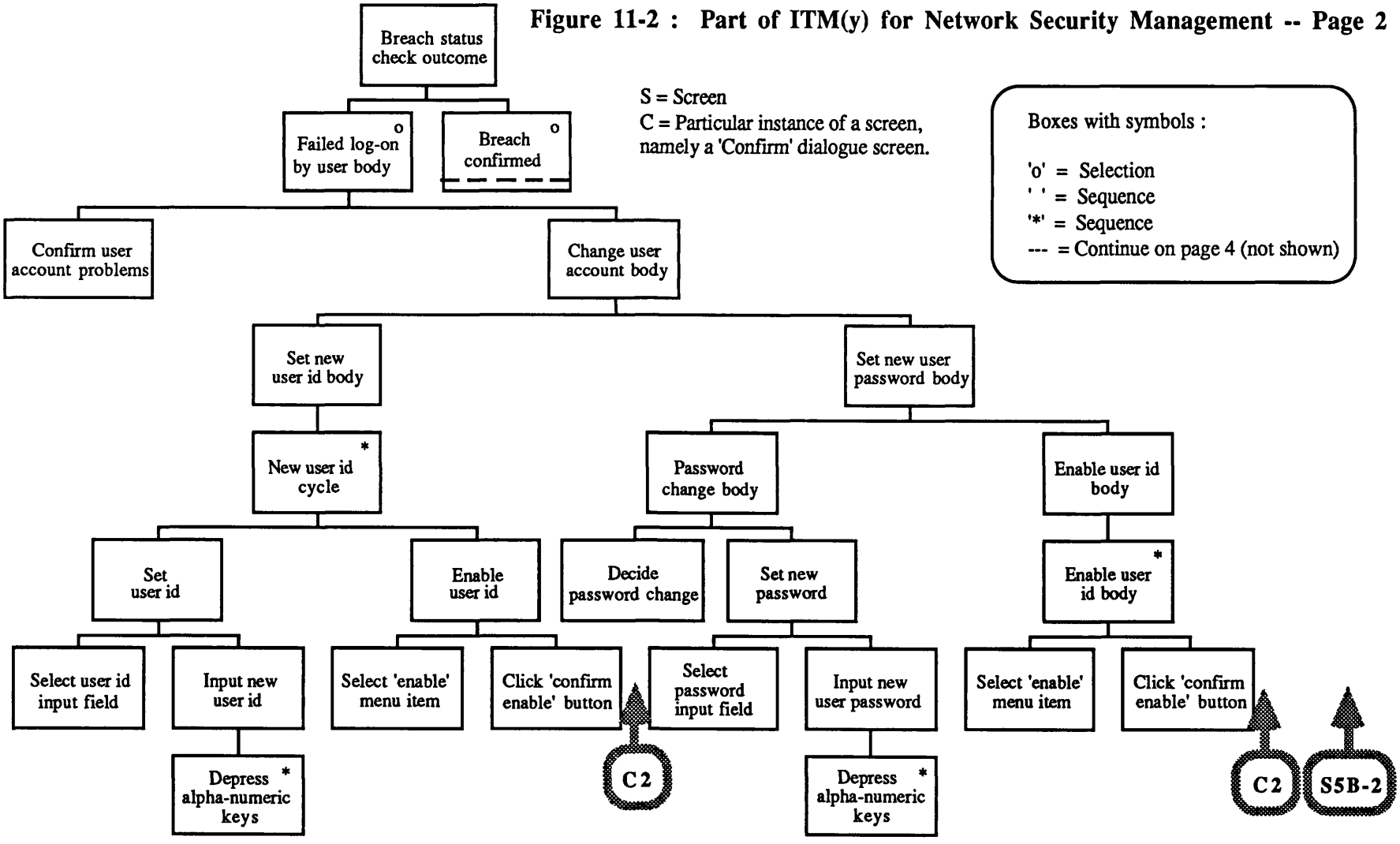
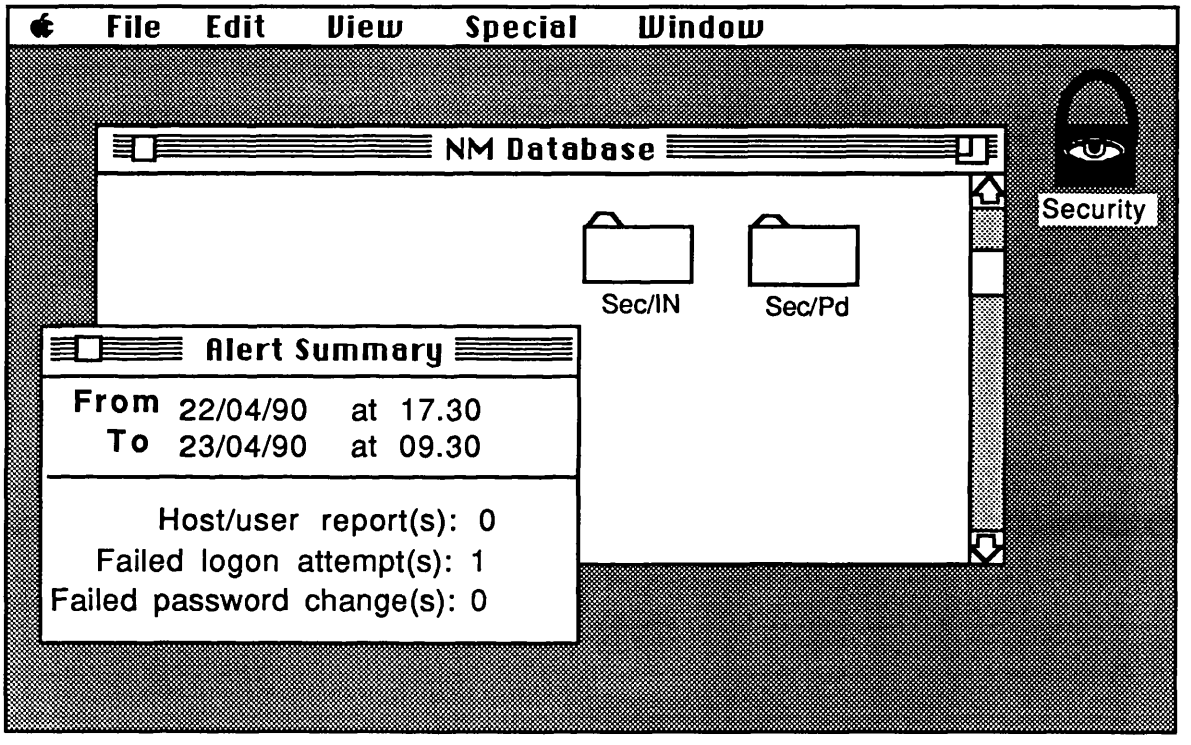


Figure 11-2 : Part of ITM(y) for Network Security Management -- Page 2



'bubbles' labelled S3A and S3B in Figure 11-2 -- Page 1). Screen 3A describes a scenario where the network management workstation is unmanned. In this case, the workstation is required to automatically monitor security breach events and user reports, and then display the number and classes of incidences on the next log-on by the network manager (see Figure 11-3).

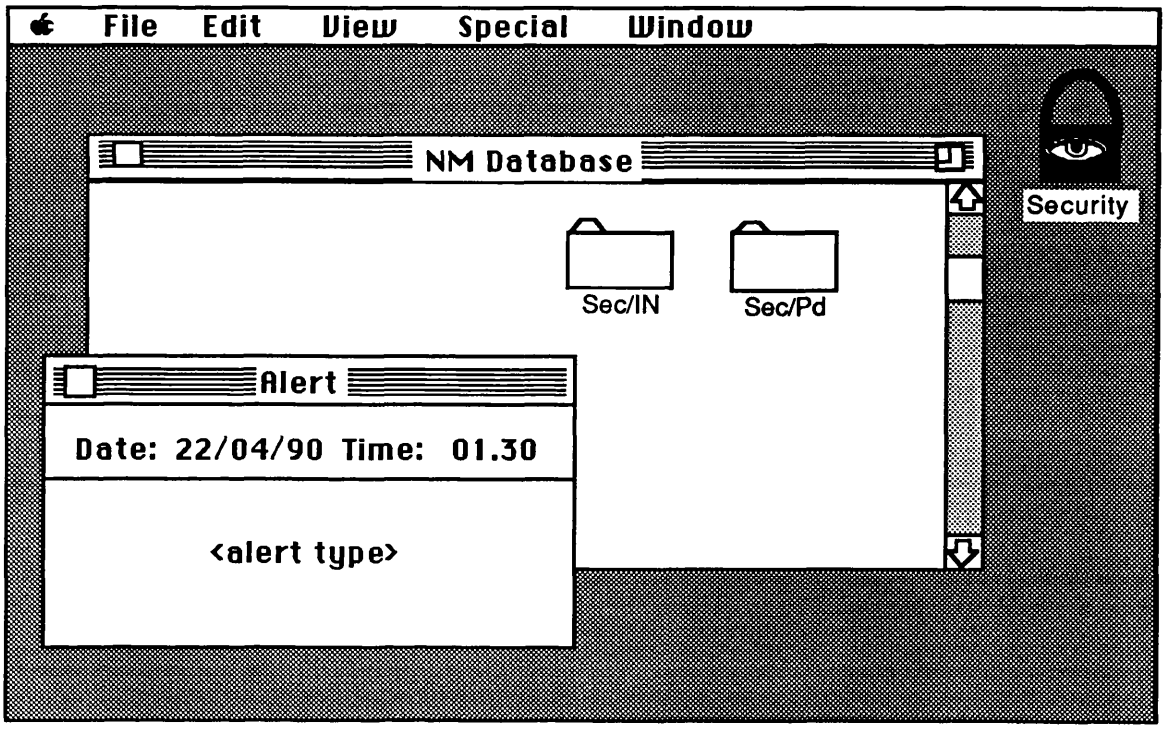
Figure 11-3 : Pictorial Screen Layout of Screen 3A



Screen 3B describes an alternative scenario for which security breach events and user reports occur when the workstation is manned (see Figure 11-4). In this instance, the workstation is required to alert the manager (immediately in certain circumstances) that a security event has occurred in the background of an ongoing interactive session.

Wider links between ITM(y) and other products of the Interface Model and Display Design Stages are addressed later in this chapter.

Figure 11-4 : Pictorial Screen Layout of Screen 3B



Having specified the device-level inputs required of the user, the composition, layout and behaviour of screen displays may now be considered. These design concerns are described in the next sub-section.

11.2. Interface Model (IM) and Display Design (DD) Stages

Summary

To ensure coherent design specification, the remaining stages of the method (namely the Interface Model and Display Design Stages) are undertaken iteratively. For this reason, they are described together in this sub-section.

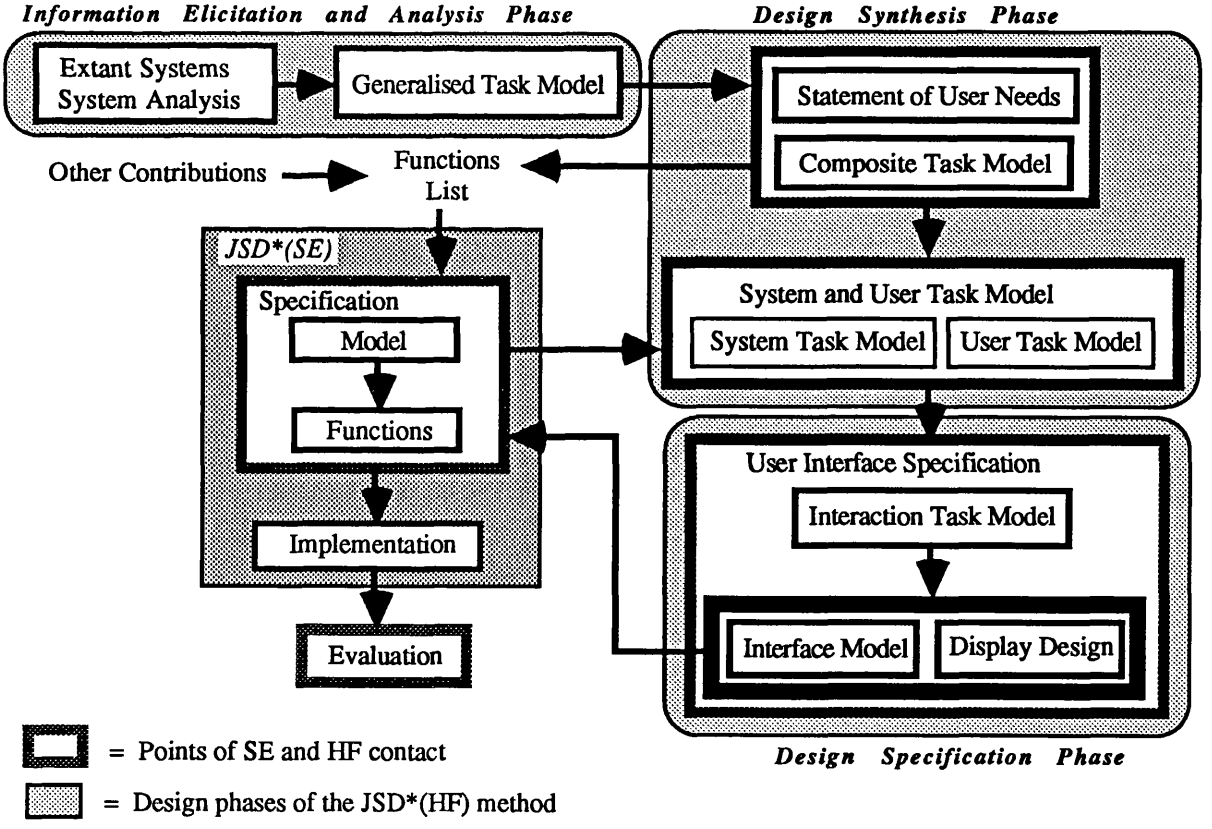
The objective of the *Interface Model Stage* is to specify the behaviour and appearance of screen objects in relation to user inputs and state changes of representation and real world entities. Thus, object modelling, command syntax and

icon design comprise the design concerns of the stage.

The objectives of the *Display Design Stage* are as follows :

(a) to specify the content and layout of display screens;
(b) to compile a glossary of screen objects; and
(c) to define the contexts for presenting error and feedback messages, and computer support functions.

The locations of the above stages vis-a-vis other stages of the JSD*(HF) method are shown in the Figure below.



As shown in the Figure, STM(y) and ITM(y) comprise primary inputs to the stages. In addition, appropriate consideration should be given to the input sub-system specifications derived earlier by JSD*(SE) analysts, and relevant sub-sets of

extant design descriptions, namely DD(ext) and IM(ext) descriptions (Note : appropriate sub-sets of the latter descriptions comprise those which are consistent with parts of the GTM(x) description that have been incorporated into CTM(y)). On the basis of the preceding products, suitable display designs are specified at the Interface Model and Display Design Stages of the method. Specifically, the preceding products constitute design requirements and constraints to be satisfied by a particular user interface design, e.g. specific screen compositions and behaviours. Thus, prototyping and user tests are particularly important at these design stages.

In summary, the set of descriptions derived at the Design Specification Phase of the method constitute HF specifications of the user interface. Subsequently, the descriptions are discussed and synthesised with JSD*(SE) specifications. Design implementation is then undertaken following the original JSD method.

A detailed account of interface model and display design descriptions follows.

Detailed Account

(a) Interface Model Specifications for the Target System (IM(y))

At the Interface Model Stage, a set of specifications (termed IM(y)) is derived to describe the behaviour and appearance of screen objects. Two categories of objects are described, namely bespoke objects and variant objects of the chosen user interface environment (if any). In most cases, generic objects of the chosen environment are not described since the design team would be conversant with their characteristics.

IM(y) descriptions are derived by *decomposing* C leaves of the STM(y) description, i.e. computer actions. The decomposition should satisfy the conditions prescribed by JSD*(HF) products that have been derived earlier, such as DoDD(y), SUN(y) and the enhanced statement of target system requirements. On this basis,

the designer may consider the following :

(1) adopting a 'global' concept to structure the user interface, e.g. the application of an appropriate user interface metaphor;
(2) porting IM(ext) descriptions that are consistent with parts of the GTM(x) description that have previously been recruited to the target system;
(3) selecting an appropriate user interface environment or house-style;
(4) introducing bespoke design extensions.

A set of IM(y) descriptions is thus derived and documented using structured and pictorial diagrams. The rationale relating to the above design decisions should also be documented as appropriate. These descriptions are then carried forward to support screen specification at the Display Design Stage. In particular, explicit relationships among IM(y), ITM(y), and screen composition and actuation are specified using an indexing scheme involving screen and object identifiers (e.g. name and appearance). Case-study examples of IM(y) descriptions are presented later.

(b) Display Design Specifications for the Target System (DD(y))

A set of JSD*(HF) products, collectively referred to as DD(y), is derived at the Display Design Stage. DD(y) descriptions address the following concerns of user interface specification :

(i) 'static' description of screen displays. Specifically, the composition and layout of information, error, feedback and help screens are specified pictorially. The descriptions (termed Pictorial Screen Layouts or PSL(y)) are supported by a Dictionary of Objects (DO(y)), and a Dialogue and Error Message Table (DET(y));
(ii) 'dynamic' description of screen displays. Specifically, the context for actuating display screens to present computer task support functions and messages is specified. The structured diagram description derived is termed a Dialogue and Inter-Task Screen Actuation Description or DITaSAD(y).

The above JSD*(HF) products are now described in more detail.

Pictorial Screen Layout (PSL(y)) diagrams describe the content, appearance, location and grouping of information and functional screen objects. Explicit relationships among PSL(y), ITM(y), DITaSAD(y) and IM(y) descriptions are specified using an indexing scheme involving screen and object identifiers (e.g. name and appearance). Thus, the static and dynamic descriptions of a screen design are inter-linked as follows :

(i) PSL(y) descriptions are linked with IM(y) descriptions at the *objects* level. Specifically, static PSL(y) descriptions are complemented by dynamic IM(y) descriptions of screen objects, e.g. their individual behaviours and their relationships with other objects (objects may reside in the same display screen (intra-screen object-object relationships) or different display screens (inter-screen object-object relationships);
(ii) PSL(y) descriptions are linked with DITaSAD(y) at the *screen* level. Specifically, static PSL(y) descriptions are complemented by dynamic DITaSAD(y) descriptions of display screens, e.g. their actuation contexts and triggers.

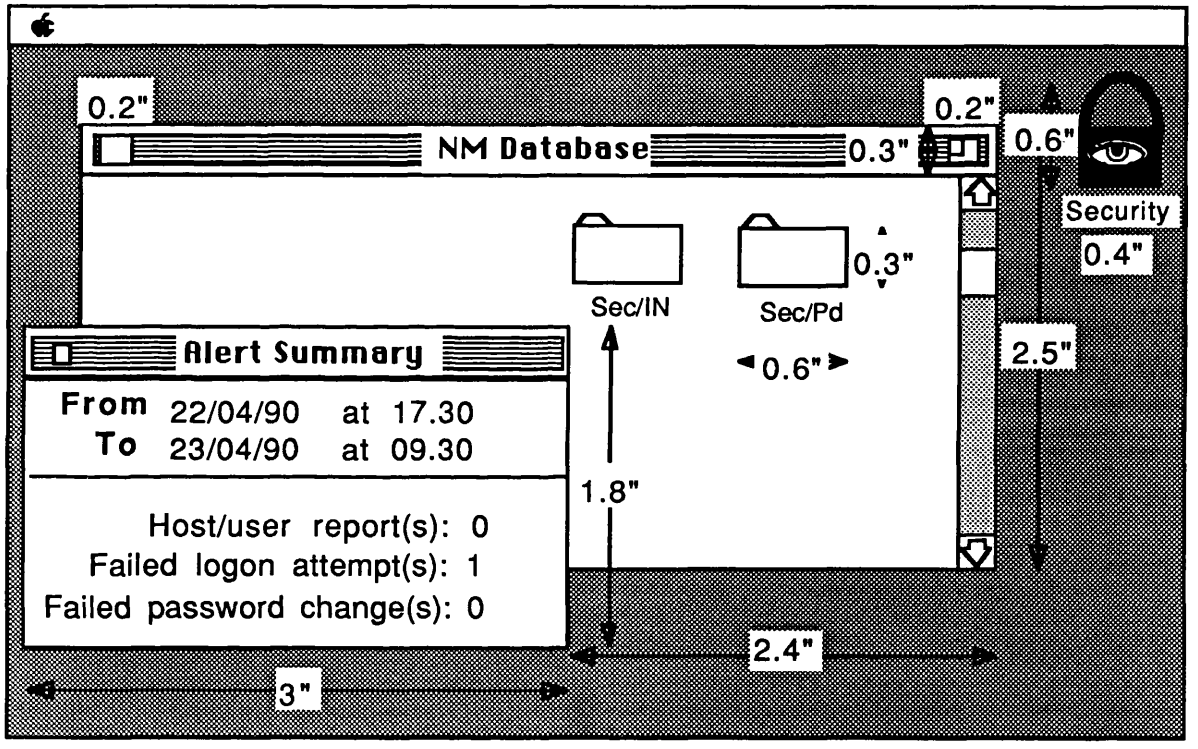
It should be noted that the objective of a PSL(y) description is <u>not</u> the specification of all possible display screens. For instance, the description of screen scrolling and refresh is excluded. Instead, the objective of a PSL(y) description is to specify the display screens associated with potential user errors and interactive task contexts. Thus, PSL(y) is inter-linked with DET(y), ITM(y) and DITaSAD(y).

PSL(y) descriptions are generally supported by DO(y) descriptions which provide further information on the objects comprising each display screen, e.g. salient characteristics such as inter-screen triggers and permissible user actions. In addition, PSL(y) descriptions of error, feedback and help message screens are supported further by a message index or DET(y) table (refer to its account on the following page).

It should be noted that PSL(y) descriptions may be composed directly using a

computer-based prototyping tool. In this way, voluminous paper-based documentation and laborious scale drawings or dimensioned screen diagrams (see Figure 11-5) may be obviated.[22] Although PSL(y) and DET(y) descriptions are thus supplanted, IM(y), DITaSAD(y) and DO(y) descriptions should still be documented as required by the method. Case-study examples of PSL(y) descriptions are presented later.

Figure 11-5 : Dimensioned Pictorial Screen Layout of Screen 3A



All fonts size 12 Geneva

The *Dialogue and Error Message Table (DET(y))* is essentially an index of message identifiers and contents. As indicated previously, the identifiers comprise part of a scheme for describing explicit links between PSL(y) and DITaSAD(y). Specifically, the contexts and triggers for presenting particular display screens are

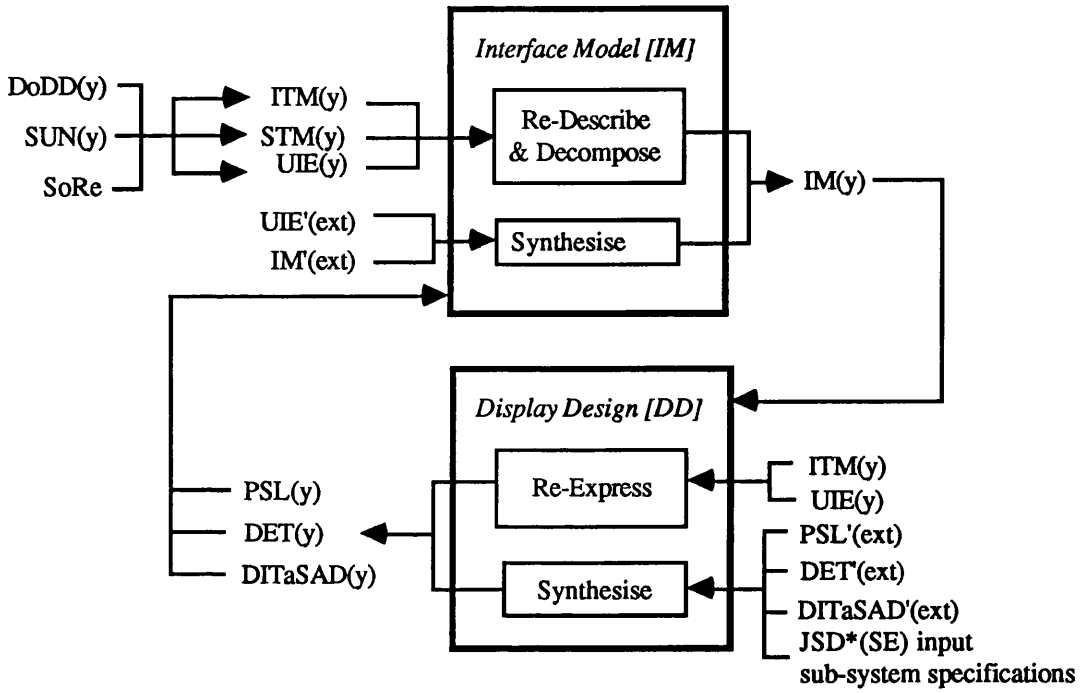
[22] Other benefits of using a prototyping tool may include directly executable specifications and animation of proposed designs.

defined with respect to the status of interactive task performance. Links between DET(y) and PSL(y) descriptions of the content and layout of screen messages are similarly established by assigning unique message and screen identifiers (refer to the earlier account of PSL(y) descriptions). A case-study example of DET(y) is presented later.

Finally, the *Display and Inter-Task Screen Actuation Description (DITaSAD(y))* sets screen actuations against the interactive task context. In other words, it specifies the points at which particular computer messages and functions are presented in support of the user's on-line task (refer to earlier accounts of PSL(y) and DET(y) descriptions). It should be noted that the objective of a DITaSAD(y) description is not the specification of all possible screen actuations. For instance, the description of screen scrolling and refresh is excluded. Instead, the objective of DITaSAD(y) is to set major screen actuations against the status of interactive task performance. In other words, it relates the actuation of display screens (described by PSL(y)) to users' interaction difficulties and errors (described by DET(y)), and to transitions between coherent interactive task units (described by ITM(y)). Thus, DITaSAD(y) is inter-linked with PSL(y), DET(y) and ITM(y). A case-study example of a structured diagram description of DITaSAD(y) is presented later.

Figure 11-6 summarises the activities and products of the design stages described above.

Figure 11-6 : Block Diagram Summary of the Interface Model (IM) and Display Design (DD) Stages



DET = Dialogue and Error Message Table

DITaSAD = Dialogue and Inter-Task Screen Actuation Description

DoDD = Domain of Design Discourse

(ext) = JSD(HF) descriptions of extant systems (EXT)*

ITM = Interaction Task Model

PSL = Pictorial Screen Layout

SoRe = Enhanced Statement of Requirements

STM = System Task Model

SUN = Statement of User Needs

UIE = User Interface Environment

(y) = JSD(HF) descriptions of the target system (Y)*

Procedures for deriving IM(y) and DD(y) descriptions are presented on the following two pages.

Procedures for deriving IM(y) and DD(y)

1. *Do not begin specifying IM(y) and DD(y) until ITM(y) is sufficiently decomposed into input level primitives. However, complete decomposition of ITM(y) is not necessary prior to the Interface Model and Display Design Stages (see (8) below).*

2. *On the basis of screen demarcations on ITM(y), a human factors designer may apply either 3(a) or (b). The choice would depend on prevailing design circumstances such as how familiar the designer is with the chosen user interface environment (if any), and how well defined the chosen environment is. PSL(y) and IM(y) descriptions are thus derived incrementally with due references to ITM(y). Note that iterations among the Interaction Task Model, Interface Model and Display Design Stages may instigate changes to design descriptions derived earlier. The process is continued iteratively until the entire ITM(y) description has been considered.*

3(a). *For each demarcated screen in ITM(y), specify IM(y) descriptions of screen objects. On the basis of these IM(y) descriptions, compose appropriate PSL(y) diagrams. Repeat the process as indicated in (2) above until the entire ITM(y) description has been considered.*

3(b). *For each demarcated screen in ITM(y), compose an initial PSL(y) diagram. Specify an IM(y) description for each object in the diagram. The result is a set of structured diagram descriptions and a tabular dictionary of objects for each PSL(y) diagram. Thus, IM(y) and PSL(y) descriptions are derived iteratively. The process is repeated as indicated in (2) above until the entire ITM(y) description has been considered.*

4. *When deriving PSL(y) diagrams, note when and how each display screen is to be actuated and the objects involved in effecting the actuation. Specifically, make notes on how each screen is triggered and consumed. Such notes would support the later construction of a DITaSAD(y) description (see (9) below).*

5. *To facilitate reference, each PSL(y) diagram should be collated with IM(y) descriptions (including object dictionary tables) of its constituent objects. Note that lower level descriptions of IM(y) may be specified. For instance, HyperCard™ tye program scripts may be specified if structured diagram descriptions are sufficiently detailed. In addition, within and between screen object-object relationships may be described and linked with PSL(y) diagrams by assigning a unique identifier to each object. Thus, the level at which IM(y) may be described is flexible, and depends largely on the training of the human factors designer.*

6. *During the derivation of IM(y) and PSL(y) descriptions, H-C leaves of STM(y) (which have been carried forward to ITM(y)) may suggest 'generic' computer support functions, e.g. standard functions of text and graphics editors. In such instances, design features of relevant off-the-shelf packages (which constitute extant partial systems) may be examined to identify potential extant objects and functions for recruitment to target system design. To this end, the selected objects and functions should be consistent with ITM(y). In addition, they should be re-named and modified (as appropriate) in accordance with the semantics defined by DoDD(y). Such considerations may instigate modifications to their behaviours and representations.*

7. *Since ITM(y) describes error-free performance only, potential user errors should now be considered. These concerns are addressed by DITaSAD(y) and DET(y) descriptions. To derive these descriptions, potential user errors are identified by examining each PSL(y) diagram. In addition, various error scenarios should be investigated analytically in accordance with IM(y), ITM(y) and the input sub-system specification generated by JSD*(SE) designers. Thus, potential deviations from the 'ideal' sequence prescribed by ITM(y) are uncovered. In the first instance, re-design to rectify anticipated user errors and difficulties is considered. If a satisfactory solution can not be found, appropriate error*

Procedures for deriving IM(y) and DD(y) (con't) :

and help messages should be composed to support the user. Thus, a DET(y) table is derived and linked to the ITM(y) description. For each message item in the table, a PSL(y) diagram is composed and labelled in accordance with the rules below. To facilitate reference, the present set of PSL(y) diagrams is collated with the set of PSL(y) and IM(y) descriptions that had been derived earlier.

8. *When PSL(y), IM(y) and DET(y) descriptions have been defined satisfactorily, a further design iteration may be undertaken (as appropriate) to 'finalise' the decomposition of ITM(y) to a device-level description. Thus, ITM(y) may be described in terms of specific actions of bespoke screen objects that have now been defined by IM(y) and PSL(y).*

9. *Having 'finalised' the descriptions in (8) above, the dynamics of screen presentation is then specified. To this end, preceding JSD*(HF) products (namely ITM(y) and IM(y)) and notes made in (4) above are consulted. Thus, a structured diagram description of DITaSAD(y) is derived to summarise how each PSL(y) screen is 'triggered' and 'consumed' with respect to the user interactions prescribed by ITM(y).*

Rules of thumb for deriving IM(y) and DD(y)

1. *The constitutents of each PSL(y) screen should be described further in a supporting table termed an objects dictionary. In this way, detailed object behaviours may be highlighted.*

2. *Whenever possible, each PSL(y) screen should be uniquely and sequentially named in accordance with the order of presentation as described by DITaSAD(y).*

3. *The names of PSL(y) screens should also indicate their relationship (as appropriate), e.g. parentage. Thus, Screens 3.1 and 3.1A are both children of Screen 3, and Screen 3.1 and 3.1A are mutually exclusive selections of screens to be presented.*

4. *The appearance and name of each object should be propagated consistently across screens. Instances of an object class may be assigned a composite name comprising a root name and a unique identifier. Such information should be noted in the objects dictionary.*

5. *IM(y) descriptions need not be derived for generic objects of the chosen user interface environment.*

6. *Error and dialogue message screens should be denoted by a composite name comprising a root name followed by '.5' and a unique message identifier linked to DET(y) and DITaSAD(y), e.g. 'Screen 1.5 -- em3'. The name reads 'error screen 1.5 with message number 3 (contents shown in DET(y) table) may be triggered following Screen 1.' An alternative scheme is to name error and confirmation screens respectively using a 'E' or 'C' letter followed by a root name and message identifier as before. Note that a '.5' is not used in this case, and the name 'Screen E(1)1 -- em3' should be read as before. However, the name also indicates that message em3 is displayed using a variant of a generic error screen template called Screen E(1). Specifically, only the <message content> of Screen E(1) is varied.*

7. *DITaSAD(y) may be specified in two or more steps to facilitate its derivation. In particular, the designer may exclude error considerations from an initial DITaSAD(y) specification, i.e. screen actuations corresponding directly to ITM(y) are described first. Error scenarios are then introduced to complete the description of DITaSAD(y).*

8. *Structured diagram leaves of a DITaSAD(y) description largely comprise 'consume screen' boxes. Thus, in deriving a DITaSAD(y) description, intervening action leaves of the ITM(y) description that do not result in a screen actuation are removed (since DITaSAD(y) (and PSL(y) descriptions) is concerned largely with major screen actuations). Nevertheless, the super-ordinate structure of ITM(y) should still be carried forward.*

Case-Study Illustrations of IM(y) and DD(y)

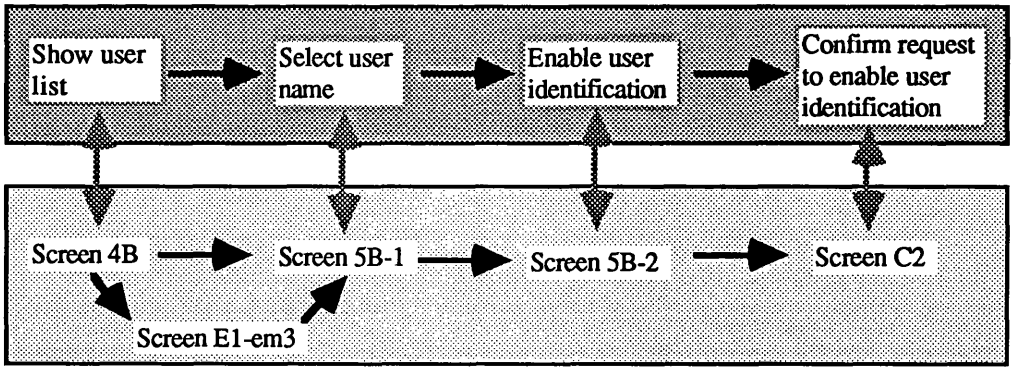
A case-study illustration of products derived at the Interface Model and Display Design Stages follows. In particular, case-study examples are described to highlight inter-linkages among the products of these stages. To this end, the case-study scenario introduced in preceding accounts of earlier JSD*(HF) products (i.e. a 'failed log-on' event) is used again for the present illustration. It may be pertinent to note that the following account describes JSD*(HF) specifications of a user interface design (a set comprising IM(y), DITaSAD(y), DET(y) and PSL(y) descriptions).

(a) Dialogue and Screen Actuation Description (DITaSAD(y)

Figure 11-7 shows part of a DITaSAD(y) that describes target system responses to a failed log-on event. Specifically, it describes the on-line component of the network manager's task in terms of a succession of major screen actuations, e.g. Screens 4B, 5B-1, etc. An information table is not derived in this instance since the structured diagram description of DITaSAD(y) is self explanatory.

Target system responses to a failed log-on event may be conceptualised as comprising two component streams, namely human and computer responses to the event. The streams are described by ITM(y) and DITaSAD(y) descriptions respectively (see Figure 11-2 (Page 1) and Figure 11-7). By specifying linkages between the streams, screen actuations (comprising the sequence : 'Screen 4B' --> either 'Screen E1-em3' or 'Screen 5B-1' --> etc.) are contextualised against the on-line tasks to be performed by a network manager (comprising the sequence : 'show user list' --> select user name' --> etc.). The inter-links are summarised in the diagram overleaf. For complete user interface specification, the above descriptions are complemented further by IM(y), PSL(y) and DET(y) descriptions. An account of these JSD*(HF) descriptions follows.

Network manager : on-line tasks following a failed log-on event



Network management workstation : screen actuations following a failed log-on event

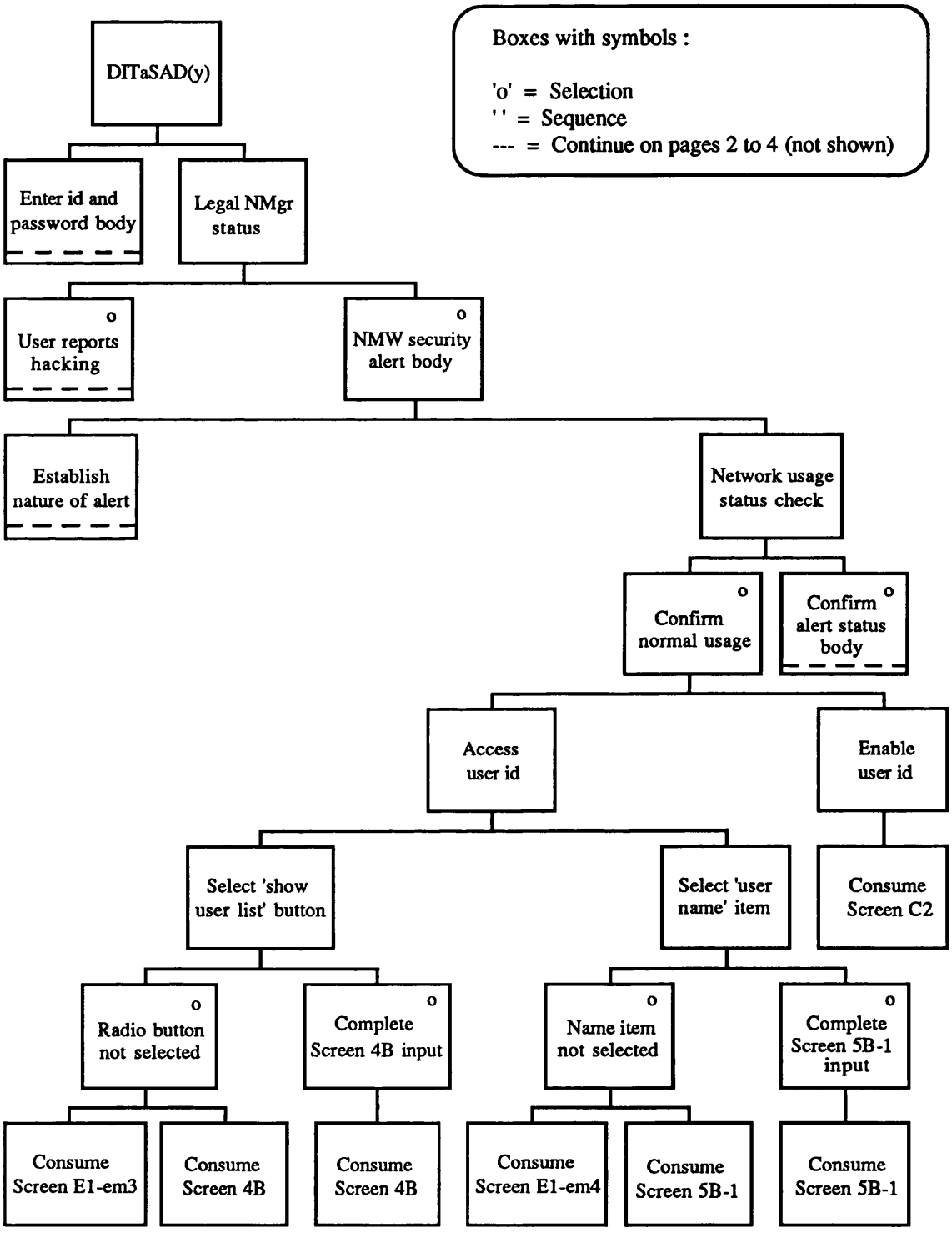
(b) Pictorial Screen Layout (PSL(y)), Interface Model (IM(y)) and Dialogue and Error Message Table (DET(y))

Following an alert to a failed log-on event (see Figures 11-3 and 11-4), it was indicated earlier that a series of screens is presented to the network manager. JSD*(HF) descriptions of the design of these screens and their constituents are illustrated below. The descriptions are collated into sets that reflect the sequence of screen presentation.

Having been alerted to a failed log-on event, the network manager is required to access the computer database so that further information may be gathered on the user involved. To this end, the manager double-clicks the security icon in Screen 3C (not shown since it is similar to Screens 3A and 3B -- see Figures 11-3 and 11-4 respectively) to activate the application for network security management. The input triggers Screen 4B, and a menu offering a selection of three actions, namely 'Search Connection', 'Show User List' and 'Show Access Points', is thus presented to the network manager (see Figure 11-8).[23] To indicate the desired selection, the manager mouse-clicks one of the three radio buttons followed by the

[23] To support the PSL(y) description of a screen, further information on its constituents is tabulated in an Objects Dictionary, e.g. the PSL(y) description of Screen 4B is supported by Table 11-1. In addition, the behaviour of individual screen objects is expanded in an IM(y) description, e.g. the IM(y) description in Figure 11-9 describes radio button objects in Screen 4B.

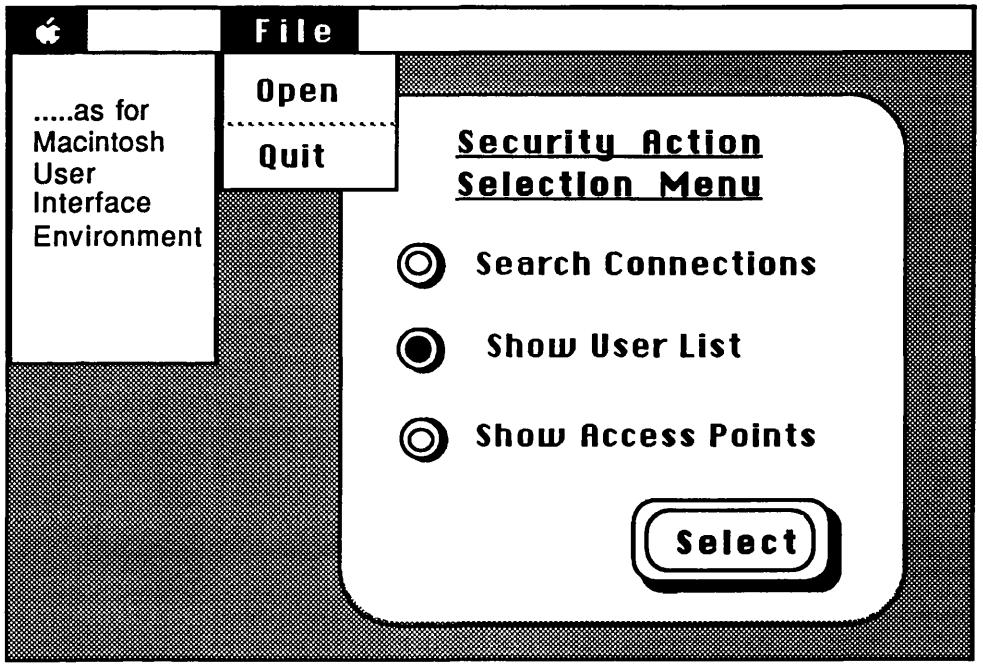
Figure 11-7 : Part of DITaSAD(y) for Network Security Management



E1, C2 = Particular instances of a screen, namely an 'Error' and 'Confirm' dialogue screen.
em = error message
NMW = Network Management Workstation
NMgr = Network Manager
S = Screen

'Select' button (see Figures 11-8 and 11-9). If either of these inputs is omitted, an error message screen, namely Screen E1-em3, is activated (see Figure 11-10 for a description of the *generic* error screen template named Screen E1, and Table 11-3 for the content of error message 3 (or em3)).[24]

Figure 11-8 : Pictorial Screen Layout of Screen 4B

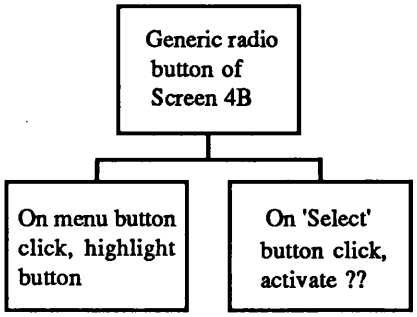


[24] In other words, error message 3 or em3 shares the same screen design as other error messages, namely Screen E1. Specifically, the screen layout is the same for these messages, i.e. the only difference is the content of their error messages.

Table 11-1 : Objects Dictionary -- Screen 4B

Screen Object	Description	Design Attributes
File (menu bar)	Offers 'Open' and 'Quit' menu items. 'Open' allows the network manager to open host and user reports. 'Quit' allows the manager to quit the security application.	Behaviour as per standard Macintosh menu items.
Security Action Selection Menu	Allows the network manager to select an appropriate action, namely 'Search Connections' (for details of network connections), 'Show User List' (for a list of network users), and 'Show Access Points' (for a list of Access points).	
Radio buttons	Allows the network manager to select an action from the menu above.	Behaviour as per standard HyperCard radio buttons.
Select button	On activation, this button activates the next screen. Depending on the network manager's selection, one of the following screens is activated : -- Search Connections : Screen 5A-1; -- Show User List : Screen 5B-1; and -- Show Access Points : Screen 5C-1.	Behaviour as per standard HyperCard buttons.

Figure 11-9 : IM(y) Description of Radio Buttons in Screen 4B



where :

?? = Screen 5A-1 if the menu button clicked is 'Search Connections'.
?? = Screen 5B-1 if the menu button clicked is 'Show User List'.
?? = Screen 5C-1 if the menu button clicked is 'Show Access Points'.

Figure 11-10 : Pictorial Screen Layout of Screen E1

Table 11-2 : Objects Dictionary -- Screen E1

Screen object	Description	Design Attributes
Dialogue box	The dialogue box is activated in response to a user input error. An error message is displayed in the box (see DET(y)).	Behaviour as per standard Macintosh dialogue boxes.
OK button	Allows the network manager to acknowledge the message and return to the previous screen.	Behaviour as per standard HyperCard button.

Table 11-3 : Part of DET(y) for the Target Network Security Management System

Message no.	Message
em1	Sorry, your log-on inputs are incorrect. Your session will be terminated.
em2	Please indicate a host and/or user report action by selecting either the 'Delete' or 'Pending' radio button.
em3	Please indicate the required security action by selecting a radio button from the 'Security Action Selection Menu'. Do this BEFORE clicking the 'Select' button.
em4	Please select a user name from the user name display window. Do this BEFORE clicking the 'Show' button.
etc.	etc.

If the inputs for Screen 4B have been made correctly, Screen 5B-1 is presented (see Figure 11-11). Using this screen, the network manager may specify what network user information should be collated and displayed as lists. For instance, the manager may request a list of user identifications that have been disabled temporarily. Following an examination of the list, the manager may access further information on a particular user by mouse-clicking the desired user name item in the 'User Name Display' window (inside the 'User List' window -- see Figure 11-11). Screen 5B-2 (Figure 11-15) is thus activated to display personal and usage information on the selected user. Having assessed the information displayed in the 'User Details' window, the manager may decide to contact the user to uncover possible causes of the failed log-on event. An appropriate action is then taken. For instance, the manager may decide to restore the user account via the 'Enable' command in the menu bar (see Figure 11-15). On detecting such a request, the network management workstation will seek final confirmation from the manager by activating Screen C2 (Figure 11-16). Although Screen C2 may be considered a redundant confirmation step, it was included to make the interaction cycle more consistent with punitive actions that may be imposed by the manager. Specifically, interaction consistency is maintained since 'Disable User' and 'Mark User' actions both require positive confirmation from the manager.

The above products of the Design Specification Phase constitutes the set of human factors specifications of a user interface design.[25] Following discussions with software engineers,[26] JSD*(HF) and JSD*(SE) specifications are integrated and

[25] It may be pertinent to add that JSD*(HF) products derived between the Generalised Task Model and Interaction Task Model Stages (inclusive), could contribute to the design of training programmes and user manuals.

[26] The final integration of JSD*(HF) and JSD*(SE) specifications should be led by software engineers. This recommendation is consistent with the current training of human factors designers and their present role in system design.

then implemented as per the original JSD method.[27] Late evaluation activities would usually follow. These activities will not be reviewed here since they are already well established and are thus excluded from the present research.[28] If necessary, the reader should refer to Long and Whitefield (1986) for information on late evaluation techniques.

---

[27] Design implementation was not considered in as much detail as design specification because the transformations entailed by JSD implementation is a well regulated and mechanistic process. In addition, the external behaviour of a JSD specification is not altered by JSD implementation (see Zave, 1984). Furthermore, it is anticipated that human factors input at this stage would be confined largely to the specification of additional feedback displays to account for cases where a longer than expected transient response time results from a particular JSD implementation.

[28] The exclusion of late evaluation is consistent with the objective of the present research, namely to address the *'too-little-too-late'* problem of human factors input to system design. As such, the JSD*(HF) method is concerned primarily with *early* and *continuous* human factors involvement in system development. In other words, the research emphasis is on defining processes and descriptions for design *analysis* and *specification* rather than design *implementation* and *evaluation*.

Figure 11-11 : Pictorial Screen Layout Diagram -- Screen 5B-1

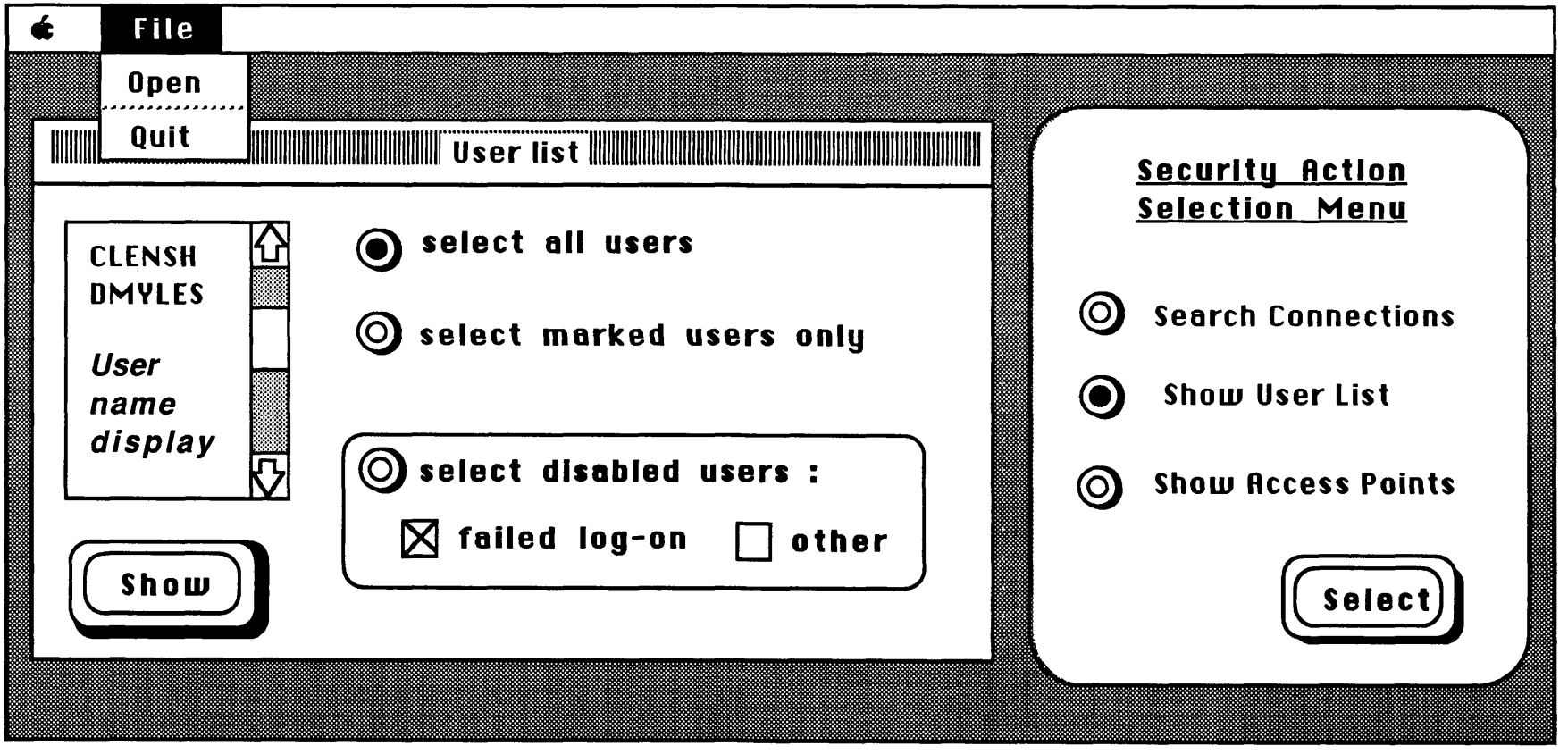


Table 11-4 : Objects Dictionary -- Screen 5B-1

Screen object	Description	Design Attributes
User list window	Used to specify particular name lists to be displayed (e.g. to display a list of disabled user names only).	Activated by selecting 'Show User List' from the 'Security Action Selection Menu' (see Screen 4B for further details).
Radio buttons	Used to specify which user names should be listed in the user name display window. Possible listings are all users, marked users only, and disabled users only (this includes two sub-lists (checkboxes in Screen 5B-1)).	Behaviour as per standard HyperCard buttons. Default display is to list all user names in the user name display. Selecting a radio button should blank out the name display, and when the 'Show' button is clicked, the specified user name list should be displayed.
User name display	Used to display user names. Activated by clicking the user list radio buttons (see above) followed by the 'Show' button.	Default display is to list all user names in alphabetical order. A user name item may then be selected (causing it to be highlighted) to display further user information in the 'User Details' window (see Screen 5B-2).
Show button	Used to trigger the display of user names following the selection of a particular user list radio button (see above).	Behaviour as per standard HyperCard buttons.
Security Action Selection Menu	Details as described previously for Screen 4B.	

Figure 11-12 : IM(y) Description of Item(s) in the 'User Name Display' Window -- Screen 5B-1

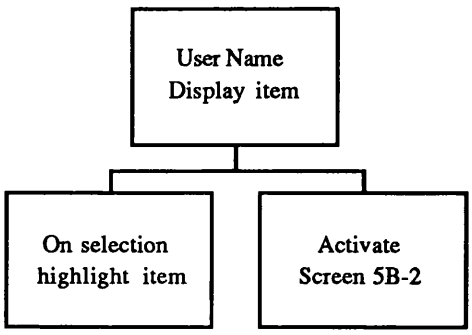
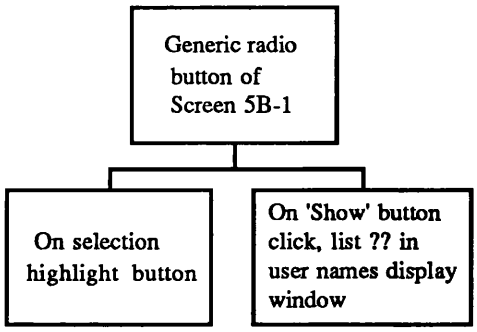


Figure 11-13 : IM(y) Description of a 'Generic' Radio Button in the 'User List' Window -- Screen 5B-1



where :

?? = complete user name list if the radio button selected is 'Select all users'.

?? = marked user name list if the radio button selected is 'Select marked users only'.

?? = complete disabled user name list if the radio button selected is 'Select disabled users' and either none or both checkboxes are selected.

?? = disabled failed log-on user name list if both 'Select disabled users' radio button and 'Failed log-on' checkbox are selected.

?? = remaining disabled user name list if both 'Select disabled users' radio button and 'Other' checkbox are selected.

Figure 11-14 : IM(y) Description of the 'Show' Button -- Screen 5B-1

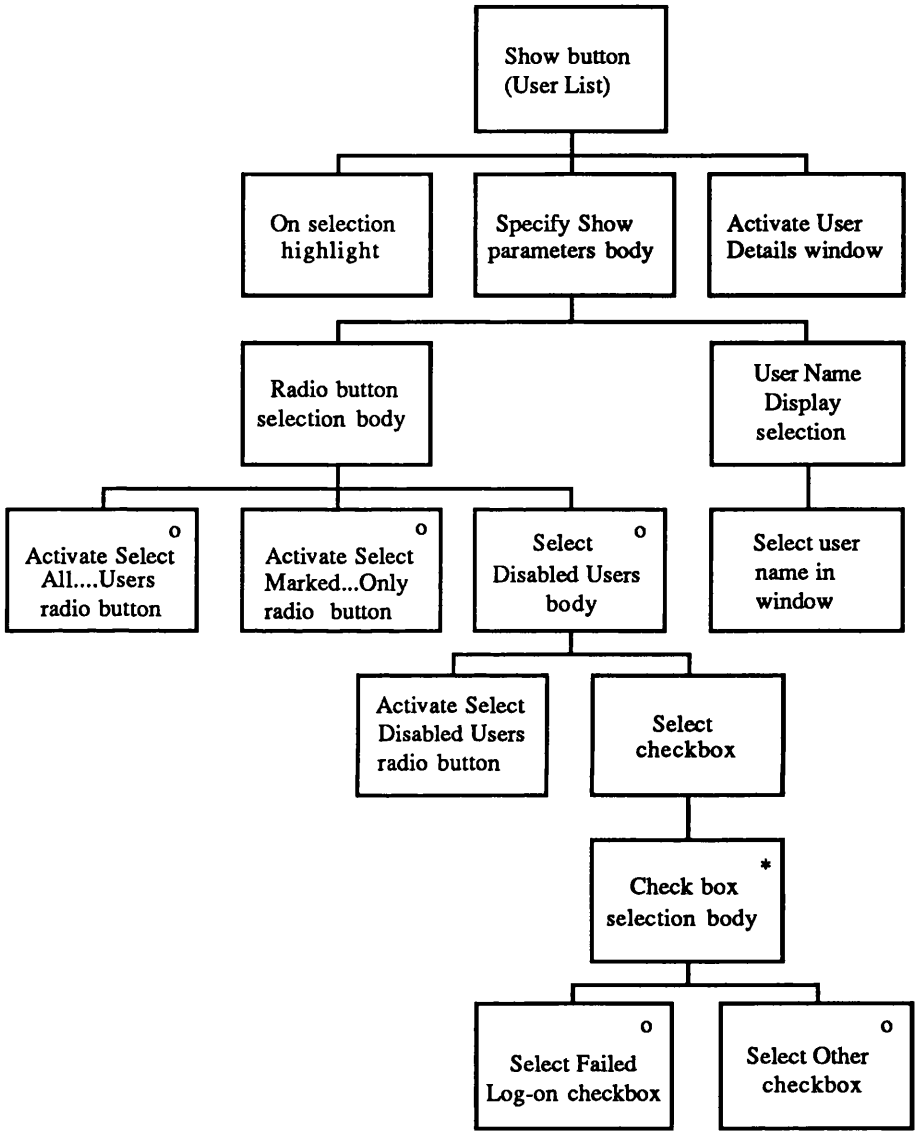


Figure 11-15 : Pictorial Screen Layout Diagram -- Screen 5B-2

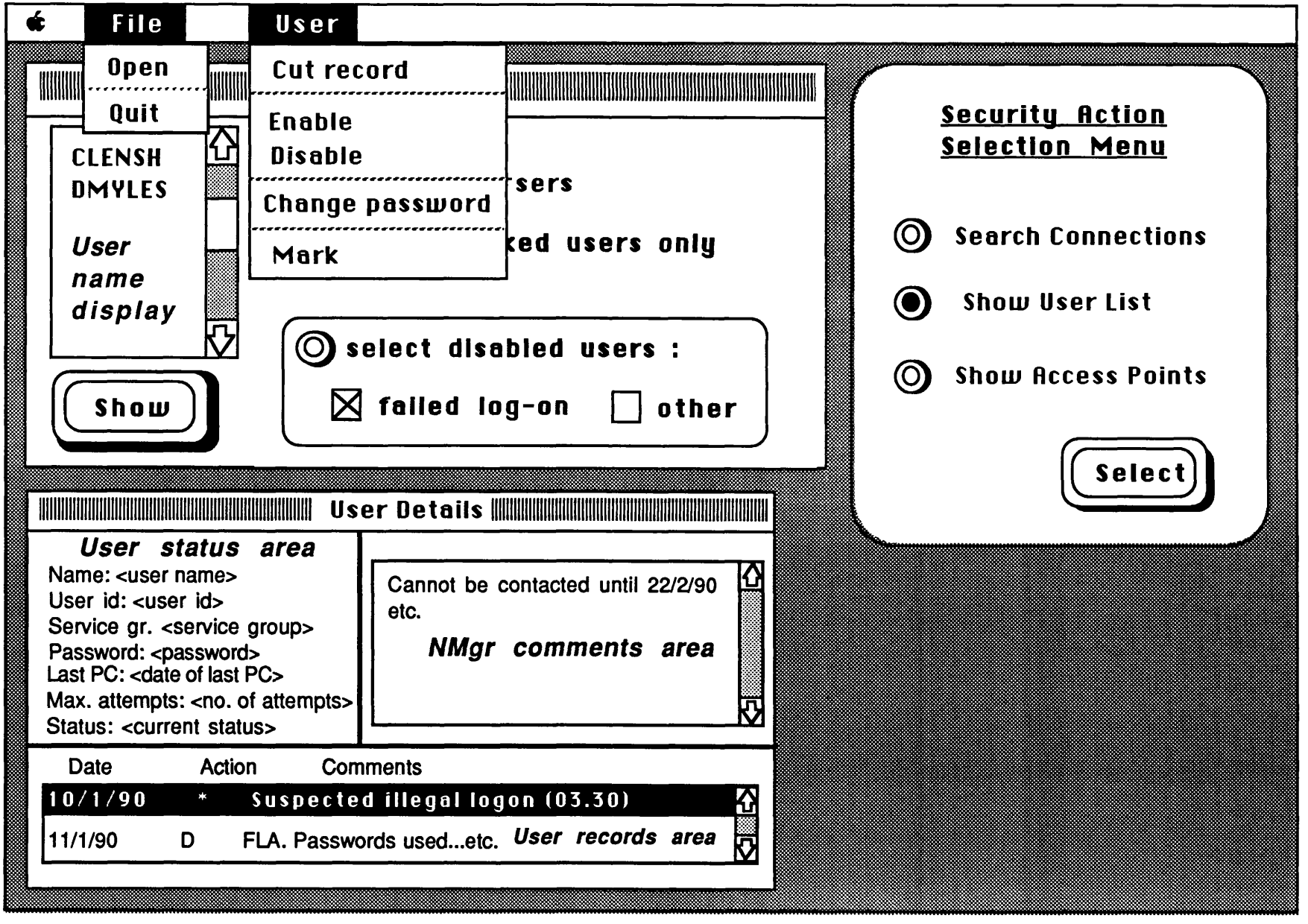


Table 11-5 : Objects Dictionary -- Screen 5B-2 (Page 1)

Screen object	Description	Design Attributes
User list window	See Screen 5B-1.	
User details window	The window displays information on a selected network user. Security actions may then be imposed on the user (mark user, password change, enable user id, etc.).	Information on the user selected in the user name display window is displayed.
User status area	Details given are as shown in the PSL(y) diagram.	With the exception of the network manager (NMgr) comments area, user details can only be edited by the network management workstation.
NMgr comments area	The network manager (NMgr) may make notes on the particular user in this text field. The notes are displayed whenever the user's details are shown, and they may be edited by the network manager.	Behaviour as per standard Macintosh text input fields.
User records area	This window displays a record of punitive actions imposed on a user id by the network manager or network management workstation. Comments made by the network manager on the punitive action is also displayed (see Screen C1). The information recorded comprises the following : (i) <date of last action>; (ii) <action> which is either disable (D), Password Change (PC) or Mark (*); (iii) <network manager's comments> made when the punitive action was taken. Mark (*) is used by the network manager to highlight suspicious user ids for closer attention. Thus, a historical record is kept on the punitive actions imposed on a user id and the reason for their imposition.	Records can be deleted but not edited. To delete a record, the network manager must click on the item and select 'Cut Record' from the menu bar (under 'User'). Records are updated following each punitive action (i.e. either disable, password change or mark). If a mark user action is imposed, an * appears in the user record. The mark may be removed by deleting the record item. If the punitive action was imposed by the network management workstation, the reason is still indicated in the comments section, e.g. user id may be disabled following a failed log-on event. The password used in the event is also recorded.
Security Action Selection Menu	See Screen 4B.	
Cut record	This command is used by the network manager to remove a 'record' from the 'User Details' window.	Available only after the record item in the 'User Details' window is selected for deletion. Behaviour as per standard Macintosh menu bar items.

Table 11-5 : Objects Dictionary -- Screen 5B-2 (Page 2)

Screen object	Description	Design Attributes
Enable	This command is used to enable a (previously disabled) user id.	Available only when the 'User Details' window is displayed (i.e. after a user name item has been selected). Activates Screen C2. Behaviour as per standard Macintosh menu bar items.
Disable	This command is used to disable a user id. Comments on the action should be recorded (see Screen C1 for more information).	Available only when the 'User Details' window is displayed (i.e. after a user name item has been selected). Activates Screen C1. Behaviour as per standard Macintosh menu bar items.
Change password	This command is used to enforce a password change on a user id. Comments on the action should be recorded (see Screen C1 for more information).	Available only when the 'User Details' window is displayed (i.e. after a user name item has been selected). Activates Screen C1. Behaviour as per standard Macintosh menu bar items.
Mark	This command is used to 'mark' a user id to highlight suspicious activities. Comments on the action should be recorded (see Screen C1 for more information).	Available only when the 'User Details' window is displayed (i.e. after a user name item has been selected). Activates Screen C1. Behaviour as per standard Macintosh menu bar items.

Figure 11-16 : Pictorial Screen Layout of Screen C2

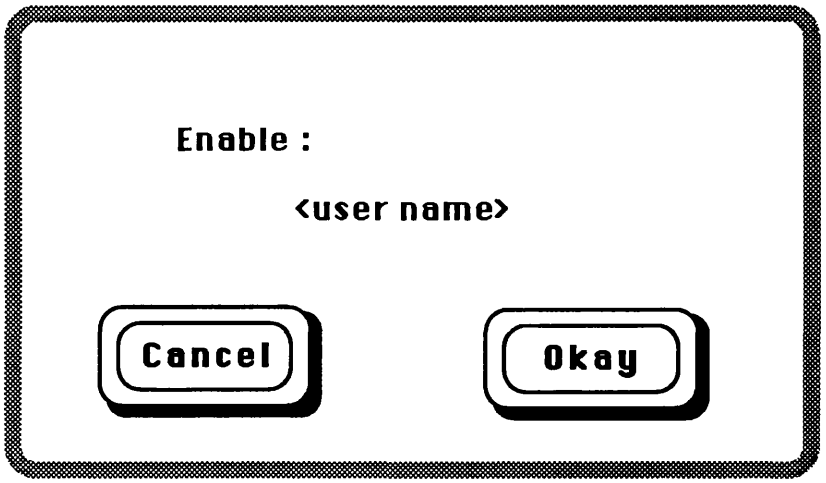


Table 11-6 : Objects Dictionary -- Screen C2

Screen object	Description	Design Attributes
Dialogue box	The dialogue box displays the user id, and confirm and cancel options after an 'Enable' action is selected. In contrast with Screen C1, comments on this action are not recorded.	The dialogue box displays the enable action and the user name in question. Behaviour as per standard Macintosh dialogue boxes.
OK and Cancel buttons	Used to confirm or cancel an 'Enable' action.	Behaviour as per standard HyperCard button.

The above account completes a review of the entire JSD*(HF) method.

In conclusion, the method contributes to system development as follows :

(a) as a structured human factors method, it specifies explicitly the scope, process and notation of human factors contributions to the entire system development cycle;
(b) as part of an integrated Human Factors and Software Engineering method (namely JSD*) :

(i) it supports timely and contextually relevant human factors input to system design;
(ii) it facilitates an explicit accommodation of human factors design needs by the overall system design agenda;
(iii) it defines clearer roles and design relationships between human factors designers and software engineers.

Consequently, a more effective uptake of human factors contributions may be expected from the application of the method.

PART V :

Synopsis of the Research

CONTENTS

Chapter Twelve : An Assessment of the Present Work and Opportunities for Follow-up Research.....309

12.1. An Assessment of the Human Factors Design Scope Addressed by the Research.....310
12.2. An Assessment of the Research Scope and Activities for Developing a Structured Human Factors Method.....313
12.3. An Assessment of the JSD*(HF) Method.....319
12.4. Wider Extensions of the Research.....330
12.5. Concluding Summary.....335

Chapter Twelve : An Assessment of the Present Work and Opportunities for Follow-Up Research

"The old order changeth, yielding place to new....."

Lord Tennyson, 1809-1892.

"The meaning of things lies not in the things themselves
but in our attitude towards them."

Antoine de Saint-Exupéry

Since this chapter concludes the thesis, its scope includes the following :

(a) assessing the human factors design scope addressed by the research. For instance, was an appropriate set of concerns addressed ?

(b) assessing the formulation and implementation of research plans and activities. For instance, were appropriate case-study systems and tests used during method development and demonstration ?

(c) assessing the research product. For instance, were research requirements satisfied fully by the proposed method, and what were its limitations ?

(d) identifying the possibilities for further extensions of the method. For instance, a wider scope of human factors integration with the JSD method may be considered in (a), (b) and (c) above;

(e) identifying further research to support methodological integration as a means of incorporating human factors into system development. For instance, computer-based tools may be developed to support the integrated method, and declarative human factors knowledge may be identified for application at various stages of the method.

An account of the above considerations follows.

12.1. An Assessment of the Human Factors Design Scope Addressed by the Research

It was observed in earlier chapters that the research scope is potentially very wide. Thus, a scope commensurate with the project resources had to be defined. After careful consideration, a concise and coherent set of human factors design concerns was identified (see Figure 7-5, Chapter Seven). However, the research undertaking was still extensive and an in-depth study of all human factors design concerns was not feasible. Thus, compromises had to be made to balance appropriately a breadth-first and depth-first coverage of the set of design concerns. Specifically, only 'key' human factors concerns were addressed in detailed. Such concerns were selected on the basis of the following criteria :

(i) the current state of human factors knowledge with respect to the particular design concern. Specifically, design concerns which are well established and supported are assigned a lower research priority, e.g. late human factors evaluation;
(ii) the human factors design support required by the SADM chosen for integration. Thus, the requirements of the JSD method were accommodated preferentially by the present research (see Chapter Six);
(iii) the research resources required to address the particular design concern. Generally, to avoid jeopardising other research requirements design concerns which require extensive resources were not studied in detail;
(iv) the potential alleviation of existing problems of human factors input that would be gained from a better accommodation of the particular design concern. Thus, the criterion pre-disposes the research towards addressing design concerns at earlier stages of system development;
(v) the potential improvement of the quality of the final artefact that would be gained from a better accommodation of the particular design concern. This criterion was not applied in isolation since it is difficult to attribute specific improvements in artefact usability and functionality to individual design concerns. Thus, the criterion was used only to support (i) above.

Bearing the above criteria in mind, the scope of design concerns addressed by the

present research (namely user requirements specification, task analysis and user interface design -- see Chapter Six) may now be assessed.

First, an in-depth study of user requirements specification was excluded from the research.[1] Instead, its component concerns, namely user requirements elicitation, analysis and specification, were addressed by the present research as follows :

(i) existing 'off-the-shelf' *requirements elicitation* techniques were recruited to the method since they were already well developed. To support the recruitment, the following pre-requisites were met :

(a) the requirements elicitation stage was located explicitly vis-a-vis other stages of the method;
(b) the subject matter and scope of requirements elicitation were defined, e.g. extant current and related systems;
(c) the design products to be derived were specified (see (iii) below);
(d) a relevant set of requirements elicitation techniques was identified.

(ii) *requirements analysis* was incorporated with task analysis since their design concerns overlap to a large extent. Thus, a number of existing task analysis techniques were recruited and incorporated into the ESSA and GTM Stages. Nevertheless, the method could benefit from a wider survey of knowledge engineering literature since more advanced requirements analysis techniques may be uncovered for recruitment, e.g. repertory grid and cluster analyses. Such surveys could be pursued in a follow-up research project;
(iii) *requirements specification* products were specified as comprising a statement of user needs table, a domain of design discourse description, and a performance specification table (i.e. products of the SUN Stage). Although case-study tests showed that design specification was supported by these products, detailed examinations of other notational schemes for

[1] The scope of user requirements specification is extensive. Thus, the resources required for an in-depth study could only be accommodated by a separate research project.

describing the domain of design discourse and for specifying system performance were not undertaken. Thus, further investigations of alternative notations may benefit the method.

To summarise, resource limitations precluded an in-depth study of user requirements specification. Instead, a comprehensive breadth-wise study was undertaken to identify well established and appropriate techniques for incorporation into the JSD*(HF) method. In this way, potentially serious limitations were obviated.

Second, the scope, process and notation of task analysis and user interface design were defined more completely and explicitly by the present research. Notable improvements over previous conceptions include the following :

(i) a structured technique for task analysis, termed extant systems analysis, was specified (see Chapter Nine). It should be emphasised that novel and variant design are both supported by the technique since task synthesis is addressed by the Generalised and Composite Task Model Stages. General advantages of the technique comprise the following :

(a) it encourages a wider consideration of alternative designs since it emphasises the analysis of *extant* systems as opposed to analysing only the *current or existing* system. Thus, 'blinkered' design may be avoided;
(b) its stage-wise procedures, inputs and products are defined explicitly to better support design;
(c) wider design concerns such as transfer of learning, training projections, definition of system performance and semantics, are also highlighted at appropriate stages of the method.

(ii) the contribution of task analysis to user interface design was exemplified completely and explicitly. In particular, to support display design extant system analysis should be followed by the derivation of system and user task models, and an interaction task model. Thus, a more complete

conception of human factors support for system design was advanced;

(iii) human factors contributions to user interface design specification were defined explicitly. In particular, human factors specifications were defined as comprising an interaction task model, and a set of interface model and display design descriptions;

(iv) a structured human factors method was developed to provide early and continuous support for system design. In other words, the scope of the method extends across requirements specification to user interface design. Thus, a comprehensive set of human factors products, procedures and notations was specified to support each stage of system design.

To conclude, the main product of the research, namely the JSD*(HF) method, is concerned predominantly with providing procedural support for human factors involvement throughout system design. Thus, it follows that subsequent research projects should identify the declarative human factors knowledge (e.g. design guidelines) that may be recruited at each stage of the method. Alternatively, follow-up projects may be concerned with specifying computer-based tools to support effective application of the method, e.g. CASE- and IPSE-type tools. Such research extensions are detailed in Sub-section 12.4.

12.2. An Assessment of the Research Scope and Activities for Developing a Structured Human Factors Method

Generally, research activities were largely implemented as planned, e.g. literature surveys; case-study selection, planning and familiarisation; specification and implementation of research strategies; specification and test of method conceptions; etc. In addition, collaborative tests of the JSD* method were co-ordinated satisfactorily except for tests involving design inter-dependencies of the method (see Sub-section 12.3). A detailed assessment of how key concerns of method development were addressed is discussed below.

(i) *case-studies* -- it was clear at project inception that the number of case-study tests would be limited by the resources available for the present

research. Thus, considerable care was devoted to the planning and selection of appropriate case-study systems and method tests. For instance, the representativeness of selected case-study systems (of its class and of system design in general) and subsequent implications for the extrapolation and generalisation of test results across system classes and design scenarios, were considered at research start-up. These considerations were necessary since they support later assessment of the capability of the JSD*(HF) method, e.g. the types of systems that may be developed using the method. Thus, it was inferred initially that both real-time and data-processing systems should be included in case-study tests of the method. However, it became clear later that such concerns were not directly relevant for the following reasons :

(a) the capability of the integrated method is determined by the capabilities of its component methods;
(b) the structured human factors method was developed with respect to a SADM. Since the SADM chosen for integration was left essentially unchanged, the capability of the integrated method would be limited by the capability of the SADM. In particular, the types of systems that may be accommodated by the integrated method are determined largely by the way design specifications are described and transformed by the chosen SADM. For instance, since JSD is an appropriate method for specifying real-time systems, JSD* (the integrated method) is expected to be similarly suitable. This expectation follows because JSD structured diagram notation is adopted by the structured human factors method and its design specifications are finally implemented following the JSD method.

Thus, system types were not considered further by the present research. Instead, the following criteria for selecting case-study systems were considered during method specification and test :

(a) system size and complexity (small to large systems, well and ill defined systems);

(b) domain of application, e.g. Recreation Booking System and Digital Network Security Management System;

(c) design scenarios, e.g. variant and new/novel design (represented by the case-study design context and the familiarity of the designer with the case-study domain); [2]

(d) user interface styles, e.g. WIMP and command-line interfaces (see Lim, 1989b).

Since the number of case-studies and method tests that could be conducted were limited, case-study systems that support the co-variation of these criteria were selected. Research strategies were then applied to manage method specification and test, and to relate the latter to the selection of case-study systems. Thus, appropriate case-study systems were identified to support each stage of method development; e.g. small and well-defined case-study systems to support the application of a backwards before forwards engineering strategy applied during the early stages of method development;

(ii) *scope of case-study tests of the JSD*(HF) method* -- generally, the objective of the case-study tests was to demonstrate the capability of the method for providing human factors support throughout the system design cycle. In contrast, the tests were not concerned with validating the efficacy of the method for ensuring superior design artefacts. This restriction of the tests was necessary because the extensive field tests required for method validation could not be accommodated by the resources available to the present research (without incurring inappropriate sacrifices to other aspects of the work). Instead, concerns involving the validity of the JSD*(HF) method were addressed during method development as follows :

(a) by developing the JSD*(HF) method in accordance with the

[2] It is unclear what constitutes 'novel design'. A satisfactory definition was not found in the literature. Thus, novel design was conceptualised as comprising a *process* involving an ill-defined design scenario, and the derivation of a new design *product* (extensions of existing designs were included, e.g. computerisation of a manual system).

requirements of a SADM. In particular, the explicit and well defined characteristics of SADMs were emulated by the JSD*(HF) method. Thus, the JSD*(HF) method, in common with all SADMs, assumes that a systematic and orderly design process would *support better* the derivation of a superior design artefact; [3]

(b) by exploiting established human factors methods and practices. Thus, the validity of the JSD*(HF) method is supported since it recruits and builds on existing human factors knowledge;

(c) by simulating Human Factors and Software Engineering contributions explicitly during case-study tests of the method. In this way, pertinent human factors support and design inter-dependencies were identified;

(d) by targeting a specific user of the JSD*(HF) method.[4] Explicit requirements were then identified and satisfied during method development, e.g. the scope and level of proceduralisation required by the method user;

(e) by testing proposed method conceptions iteratively using more than one case-study system. Thus, the JSD*(HF) method was derived only after several specification-and-test cycles and case-study systems;

(f) by targeting the official version of the JSD method. Thus, pertinent human factors support and design inter-dependencies were identified and accommodated by the JSD*(HF) method.

A more detailed account of the measures taken to address validity concerns

---

[3] It should be noted that a method alone cannot *guarantee* a superior design artefact.

[4] Specifically, the method is targeted at human factors designers with a working knowledge of the JSD method. During method development, the method user was simulated by a member of the RARDE project team. Since her knowledge of the JSD method was greater than most human factors designers, care was taken to constrain her application of such expertise. The role of a method user was later assumed (implicitly) by another team member when the original simulator went on extended leave. Since the second simulator had to learn the method from scratch in a short time (having joined the project at mid term), his experiences were indicative of the usability of the method (see Sub-section 12.3).

may be found in Sub-sections 3.3 and 6.5 (Chapters Three and Six respectively). It suffices to say here that the above measures were implemented satisfactorily during the development of the JSD*(HF) method.

Since direct validation of the JSD*(HF) method was not undertaken, it follows that future research should focus on testing the method in the field. The acceptability and efficacy of the method may thus be assessed. Assessments of interest would include the following :

(a) social implications of introducing the method, e.g. impact on existing design team relationships and practices; impact on work practices of individual designers (both JSD analysts and human factors designers);
(b) 'real world' benefits that would be gained by applying the method, e.g. cost-benefit analysis.

The results of such studies would help to identify methodological enhancements required to improve uptake of the method;
(iii) *nature of case-study tests of the JSD*(HF) method* -- generally, case-study tests could be configured to assess the functionality and usability of the method. An account of each of these assessments follows.

First, functionality assessments comprise determination of the design support provided by the method. In the context of a structured method, such assessments entail an examination of how well the design task (e.g. design management, reasoning, documentation, etc.) is supported by the stage-wise design scope, process and notation of a method. In the case of an *integrated* Software Engineering and Human Factors method (of which the JSD* method is an instance), the assessment should include an examination of the design inter-dependencies required by the method. For instance, the specified design inter-dependencies should be assessed on how appropriate and complete they are for supporting collaborative design.

Generally, functionality assessments constitute a basic part of the present research. Thus, case-study assessments were completed as planned with one exception, namely tests on the design inter-dependencies of the method were not implemented satisfactorily (a result of the difficulties in co-ordinating case-study tests involving the sponsors of the research). Nevertheless, the implications for the JSD*(HF) method were not serious (see Chapter Seven, Sub-section 7.1.3). Thus, functionality assessments of the JSD* and JSD*(HF) methods were generally positive. A more detailed account of the assessments is presented in Sub-section 12.3 where the JSD*(HF) method is discussed.

Second, usability assessments comprise determination of the *cost incurred* in applying the JSD*(HF) method. An extended assessment may include *opportunity costs* to account for benefits foregone by method application, or benefits that would be gained if alternative design approaches were applied, e.g. rapid prototyping. Thus, usability assessment involves a cost-benefit analysis of the utility of the method, e.g. benefits of method application are compared with efforts expended in learning the method *(learnability),* with having to change current work practices *(acceptability),* etc. Such assessments are indicative of the method's capacity for accommodating different users,[5] design scenarios and project characteristics, i.e. an indication of the *flexibility* and *tailorability* of the method. Since these method characteristics comprise important determinants of its uptake, they were accommodated appropriately during method development, e.g. existing methods were recruited to maximise positive transfer of learning; the chosen SADM was maintained largely unchanged; a flexible level of extant system analysis was specified; the method was characterised at different levels of description to support varying degrees of method application; etc. Unfortunately, an overall usability assessment of the method could not be supported due to the limited resources available to the present research. However, an informal assessment of the learnability of the method was afforded when a new researcher joined the project at mid-

[5] The JSD*(HF) method is targeted at a particular method user (see Footnote 4).

term (see Footnote 4). The assessment is discussed in the next sub-section.

In summary, the scope of the research comprises the development of the JSD*(HF) method and case-study assessments of its functionality. By constraining appropriately the scope of method development and assessment, the research activities were largely completed as planned. Thus, key research concerns were addressed and a 'reasonably valid' JSD*(HF) method was developed. A detailed account of an assessment of the method is discussed below.

12.3. An Assessment of the JSD*(HF) Method

In Part IV, a structured human factors method, termed the JSD*(HF) method, was described explicitly as follows :

(a) its stage-wise design scope was defined explicitly as comprising a set of products;
(b) its stage-wise design process was defined explicitly as comprising a set of procedures, rules of thumb and design inter-dependencies;
(c) its stage-wise design notation was defined explicitly as comprising a set of documentation schemes.

On the basis of these methodological characteristics, the JSD*(HF) and JSD methods were integrated to generate the JSD* method.

In general, case-study assessments of the JSD*(HF) method indicated that the requirements of structured integration were largely satisfied. Case-study assessments were directed only at the JSD*(HF) method and its design inter-dependencies, since the JSD method is essentially unchanged. These assessments are presently reviewed.

The JSD*(HF) method may be assessed on the following :

(I) its solution to current problems of human factors input into system

development. For instance, to what extent does the method alleviate existing problems of human factors input, and what are its limitations ?

(II) its potential in relation to other human factors methods, i.e. a comparative assessment involving *existing human factors* methods. For instance, how does the JSD*(HF) method compare with existing (non-structured) human factors methods ?

(III) its potential in relation to other integrated methods, i.e. a comparative assessment involving *existing integrated* methods. For instance, how well is the JSD*(HF) method integrated with JSD, and how does it compare with other integrated methods; e.g. those methods proposed by Sutcliffe and Wang (1991), Blyth and Hakiel (1989), etc. ?

(IV) its methodological characteristics in relation to the requirements of a SADM. For instance, are the stage-wise scope, process and notation of the method defined adequately for a structured method ?

It should be noted that the above assessments may intersect one another, i.e. they are not mutually exclusive. For instance, assessment (III) is a composite of assessments (I), (II) and (IV) above. A more detailed account of the assessments follows.

First, the assessments in (I) above follows from the arguments for structured integration of Human Factors and Software Engineering methods described in Chapter Two, e.g. the 'too-little-too-late' problem of human factors input; encroachment of resources for human factors design due to poor project planning or its exclusion from the design agenda; etc. Generally, case-study assessments indicated the method to be promising. In particular, its structured characteristics and complete coverage of the design cycle provide an explicit address of human factors contributions to system development. On this basis, inter-dependencies between Human Factors and Software Engineering design were specified to ensure human factors inputs that are timely and contextually relevant. Thus, the uptake of human factors contributions may be enhanced by methodological integration.

The utility of the method was also supported by post-hoc observations of a user interface design specified using rapid prototyping. In particular, some of the design

pitfalls arising from specific inadequacies of a rapid prototyping approach, could have been avoided if the JSD*(HF) method had been applied (see Lim, 1991). These assertions were illustrated by demonstrating how the user interface would have been designed using the JSD*(HF) method. However, it should be emphasised that since controlled studies were not undertaken such evidence is at best circumstantial. Consequently, controlled studies and field trials should assume priority in a follow-up research project.

Second, the assessments in (II) are related to the arguments that a structured human factors method is a pre-requisite for integration with SADMs (see Chapter Three). Specifically, it was observed that existing human factors methods provide only incomplete and implicit coverage of the system design cycle. By definition, such inadequacies would be addressed directly by the characteristics of a structured method. This requirement was satisfied by the JSD*(HF) method (see Chapter Seven). In particular, the method addresses explicitly system design concerns spanning user requirements to display design. Thus, its scope, process and notation extend significantly the design support provided by existing human factors methods. Other benefits that would be gained from its integration with JSD were discussed in (I) above.

Third, the assessments in (III) may be related to the review and critique of previous reports of human factors integration with SADMs (see Chapter Five). Specifically, the JSD*(HF) method may be assessed comparatively in terms of its methodological configuration and design support. Such a comparison was made in respect of the following assessments :

> (a) to what extent were inadequacies of other integrated methods addressed by the JSD*(HF) method ?
> (b) to what extent was the design support provided by other integrated methods matched or exceeded by the JSD*(HF) method ?

Since the JSD*(HF) method was only developed recently, an extended survey of method users was not possible. Thus, as an interim assessment, the integrated methods reviewed in Chapter Five were compared and rated by the present author.

The subjective scores for these methods are shown in Table 12-1. An inspection of the scores reveals that the JSD*(HF) method was rated highly. The outcome was expected for the following reasons :

(a) the development of the JSD*(HF) method benefited from preceding attempts at methodological integration. For instance, by building on the knowledge generated in preceding attempts, similar pitfalls were avoided in the present research, e.g. the need to define specific human factors inputs and products for each stage of the method;
(b) some preceding attempts were not concerned specifically with the development of a structured human factors method;
(c) some preceding attempts were not concerned specifically with the integration of human factors with SADMs;
(d) some preceding attempts were incidental or comprise only a part of other commercial work. Thus, resources for method development were rather limited. In contrast, seven person-years were allocated for the development of the JSD*(HF) method;
(e) some preceding attempts recruited earlier outputs of the *present* research, e.g. Sutcliffe and Wang (1991); Blyth and Hakiel (1989). Since then, these outputs have been developed further or superseded.

Fourth, it was stated in Sub-section 12.2 that the assessments in (IV) are concerned with functionality and usability assessments of the method. A detailed account of these assessments follows.

Generally, functionality assessments address the following :

(a) the design *scope* of the method. For instance, to what extent is the scope of human factors design addressed by the stages and products of the JSD*(HF) method ?
(b) the design *process* of the method. For instance, how logically sequenced, manageable and complete are the design stages of the JSD*(HF) method ? Are the design procedures of the method sufficiently explicit and complete to support design specification by the targeted user of the method ?

Table 12-1 : Comparing JSD* Against Other Integrated Methods

Integration of Human Factors with :	*Structuredness of Human Factors Method*				*Integration of Methods*		*Overall Score (Average)*	*Comments*
	Scope	*Process*	*Procedures*	*Notation*	*Common Notation*	*Design Inter-Dependencies*		
SASD Hakiel and Blyth (1990a and b); Blyth and Hakiel (1989)	3	3	3	2	1	1	2	Extensive reliance on existing human factors techniques. The suitability of these techniques was not demonstrated. Incomplete illustration of the human factors method. Method description and proceduralisation were severely lacking for later design stages. Explicit integration of SASD and the human factors method was not attempted.
SSADM Damodaran et al (1988); Ip et al (1990)	3	3	2	3	1	2	2	Extensive reliance on existing human factors techniques. The suitability of these techniques was not demonstrated. Published examples illustrate only a small part of the human factors method. Checklists were used frequently but the adequacy of their procedures is uncertain. Inadequate attempt at integrating SSADM and the human factors method.
	(Assessment based on largely unsubstantiated claims by the authors of the method.)							
JSD Sutcliffe (1988a and b); Sutcliffe et al (1991)	4	4	2	3	6	3	4	Use of CCT was unsuccessful. Inadequate definition and description of the stage-wise products, process and notation of the human factors method. Design procedures were not specified. Extensive use of a common notation. However, design inter-dependencies were not defined.
	(Assessment based on the enhanced version of the method reported in 1991.)							
JSD Carver et al (1987); Carver (1988)	2	2	2	4	6	3	3	Limited coverage of human factors design concerns. Good use of JSD notation for task description. However, explicit description and documentation of other human factors products were not addressed, e.g. domain semantics and design rationale. Inadequate specification of design inter-dependencies between JSD and the human factors method.
JSD Lim et al (present research)	5	5	5	5	6	4	5	Scope and process of human factors design concerns could be extended further. Notational rules could be specified more tightly. Design inter-dependency specifications require further substantiation. Final integration of human factors and JSD specifications should be investigated further.

1 (worst) ⟶ 6 (best)

(c) the design *notation* of the method. For instance, to what degree are requirements for comprehensive and specific human factors design descriptions supported by the stage-wise notations and documentation schemes of the method ? Are the notations and descriptions of the method adequately powerful and comprehensive to support effective communication among human factors designers, software engineers and end-users ?

(d) the design *inter-dependencies* of the method. For instance, are intersecting design concerns identified correctly and completely ? Are design inter-dependencies specified adequately to support effective and efficient collaboration between JSD*(HF) and JSD*(SE) design, i.e. to ensure design convergence without fortuitous iterations ?

In general, case-study results in respect of these assessments of the method were positive. However, the results for assessment (d) should be considered *provisionally* positive since they were inferred largely from post-hoc inspections of design descriptions derived by applying the JSD*(HF) and JSD*(SE) methods (see Chapter Seven). In particular, design inter-dependencies should be specified when the design information to be used is common to both methods, and when intersections are found between Human Factors and Software Engineering design descriptions.

The assessments also highlighted potential areas for follow-up research, namely :

(a) *design inter-dependencies of the method.* It was noted Sub-sections 7.1.3 and 12.2 (Chapters Seven and Twelve respectively) that case-study tests on design inter-dependencies of the method were not realised completely. In particular, 'real time' co-ordination between JSD*(HF) and JSD*(SE) design did not materialise during the case-study tests (see Sub-section 12.2). Thus, case-study assessments of JSD*(HF) design inter-dependencies were based largely on post-hoc comparisons of JSD*(HF) and JSD*(SE) design descriptions. As a result, more specific design inter-dependencies could not be defined to ensure a tighter synchronisation of information exchanges between JSD*(HF) and JSD*(SE) stages. Thus, follow-up research projects should repeat case-study tests involving 'real

time' co-ordination between JSD*(HF) and JSD*(SE) design.[6] Further areas of investigation concerning design inter-dependencies of the method are reviewed below.

First, an area of study not pursued in the present research concerns the determination of more explicit relationships between JSD*(HF) and JSD*(SE) design products. For instance, it is presently unclear how JSD*(HF) products may be influenced by specific specifications of the JSD*(SE) Functions Stage, e.g. types of data stream merges; time grain markers; etc. Such relationships should be identified so that appropriate design inter-dependencies may be specified. Thus, the large and unwieldy design transformations at the Functions Stage (see Finkelstein and Potts, 1985) may be supported better.

Second, the effectiveness of the Functions List in ensuring design convergence should be determined. Thus, the current definition of the Functions List may be enhanced, e.g. a clearer specification of its scope and derivation procedures. Alternatively, further design inter-dependencies may be specified to provide better support for parallel design development by JSD*(HF) and JSD*(SE) designers (see next paragraph).

Third, the potential of performing early stages of JSD*(HF) design in parallel with JSD*(SE) Model and Function Stages should be examined further. Presently, a conservative application of the method would comprise a sequential performance of the stages as follows :

(1) JSD*(HF) design stages prior to the Composite Task Model Stage are performed to completion;
(2) JSD*(HF) design is suspended at the Composite Task Model Stage;
(3) JSD*(SE) Model and Function Stages are performed to

[6] Such a follow-up project has been commissioned by RARDE (sponsors of the present research).

completion;

(4) JSD*(HF) and JSD*(SE) design is resumed in parallel.

In this way, a converging JSD*(HF) and JSD*(SE) design may be ensured with greater confidence (thus the design is efficiently managed). However, such a conservative application of the method would incur costs that may be unacceptable. For instance, human factors input to the functional definition of the target system would be delayed. As a result, an appropriate accommodation of its contributions may be hampered by constraints imposed by preceding JSD*(SE) design products. Alternatively, the efficiency promised by conservative method application would be eroded by additional iterative cycles required to accommodate late human factors input. Strictly sequential application may also complicate design management across system development projects. For instance, to avoid leaving human factors resources 'idle' at particular stages of the design cycle, a number of design projects may be inter-leaved. This work practice would be stressful if a larger number of projects are involved. In addition, design continuity would be disrupted. To resolve such issues, the design inter-dependencies of the method would have to be examined further to determine the extent to which early JSD*(HF) and JSD(SE) stages may be undertaken in parallel.

(b) *final integration of JSD*(HF) and JSD*(SE) design specifications.* Presently, the method requires the following :

(1) JSD*(HF) specifications of the user interface should be discussed with JSD*(SE) analysts;

(2) JSD*(HF) and JSD*(SE) specifications are integrated and then implemented by JSD*(SE) analysts. In other words, the analysts are required to generate an overall set of specifications so that JSD implementation rules may be applied.

To this end, explicit relationships between JSD*(SE) and JSD*(HF) specifications have to be established. The task is supported by the following rules of thumb (recruited to the method from Carver and Cameron, 1987;

and Carver, Clenshaw et al, 1987) :

(1) active user interface objects should be linked communicatively with JSD*(SE) model and function processes;
(2) one or more user actions at the user interface may comprise one input to JSD*(SE) function processes, i.e. the relationship comprises a many-to-one mapping;
(3) one or more on-line task actions of the user may correspond to one action of the JSD*(SE) model, i.e. the relationship comprises a many-to-one mapping;
(4) user interface message and display objects should be linked communicatively with JSD*(SE) information functions.

Note that these rules of thumb may intersect (i.e. are not mutually exclusive). In particular, the scope of rules (1) and (2) intersects with rules (4) and (3) respectively.

On the basis of these rules, high level relationships between JSD*(SE) and JSD*(HF) specifications were inferred, namely rules (2) and (3) above relate JSD*(SE) specifications to ITM(y) specifications; [7] while rules (1) and (4) relate JSD*(SE) specifications to IM(y) and DD(y) specifications. Since the participation of the research sponsors in collaborative case-study tests did not include the delivery of complete JSD*(SE) specifications and the final integration of JSD*(SE) and JSD*(HF) specifications, lower level relationships between the two sets of specifications could not be investigated. For instance, it is presently unclear how ITM(y) actions may be related and linked to JSD*(SE) model and function processes. Consequently, follow-up research projects should include case-study tests involving the comparison and integration of complete JSD*(SE) and JSD*(HF) design specifications. Such studies would be instrumental in

[7] The rules also relate JSD*(SE) specifications to STM(y), an intermediate design *description* excluded from the final set of JSD*(HF) user interface *specifications*.

enhancing the JSD* method as follows :

(1) by uncovering more explicit requirements for JSD*(SE) and JSD*(HF) collaboration, additional and/or better procedures may be specified (if necessary) to enhance the support provided by the JSD* method;

(2) by extending the scope of JSD*(SE) and JSD*(HF) collaboration (hence the scope of the JSD* method) to include the JSD implementation stage, human factors support may be broadened to include the translation of JSD*(HF) specifications into executable code (see also (c) below);

(3) by understanding the processes involved in the final integration of JSD*(HF) and JSD*(SE) specifications, CASE-type tools may be developed to support the JSD* method;

(4) by understanding the implications of late design modifications, consequential changes to JSD*(HF) and JSD*(SE) descriptions and specifications may be managed better. In addition, by providing explicit methodological support to facilitate a wider consideration of JSD*(HF) and JSD*(SE) information, it may be expected that proposed design modifications would be more appropriate. Thus, future studies should consider tracing the effects of design modifications on stage-wise products of the JSD*(HF) and JSD*(SE) methods.

(c) *implementation of integrated JSD* specifications*. Presently, human factors support for JSD* implementation relates largely to the transient response times of the target system. In particular, unacceptably long response times would either require an alternative JSD implementation, or additional feedback cues to the user. Since JSD implementation was outside the remit of the present research, it was not addressed in detail. As such, it follows that future studies should investigate the possibility of providing more extensive human factors support for JSD implementation.

The above review completes an account on the functionality assessment of the

JSD*(HF) method.

As regards usability assessments of the method, none were originally planned for the present research. However, an opportunity for an informal assessment was exploited when a new research team member joined the project at mid-term. An informal assessment of the learnability of the method was thus afforded. Since the new member assimilated the method within a short period of time, it was concluded provisionally that its learnability was good. A more intensive assessment would be necessary to affirm this conclusion. Such assessments should be undertaken in a follow-up research project. Another usability assessment involves a cost-benefit analysis of method application. In this respect, it may be expected that resources for design analysis and documentation would increase.[8] However, this increase does not necessarily imply an overall increase in the total resource cost. In particular, a higher expenditure of resources on design analysis and documentation may be off-set by subsequent savings due to greater efficiency in design planning, management, specification and maintenance (i.e. arguments that support the application of SADMs in general). More cannot be said at present since the collection of quantitative data is not yet possible (the method having only just been developed). Thus, to support stronger claims concerning the usability of the method, further research would be required. In particular, field studies could be conducted to examine the following :

(1) the 'real world' performance of the JSD*(HF) and JSD* methods. For instance, how well do JSD*(SE) and JSD*(HF) designers collaborate under 'real' design situations ? Other related questions would include : how well does the method support design team interaction; how much disruption to current design practice would result from introducing the JSD* method; to what extent would design creativity be constrained by the methods; etc. ?

(2) the cost-benefit profile of the JSD*(HF) and JSD* methods. To this end, both qualitative and quantitative data should be gathered from case-study applications of the methods, e.g. to what degree do the methods affect

[8] In respect of design documentation, the increase in resource costs could be reduced considerably by computer-based tool support.

system development resources and schedules; to what extent does the JSD*(HF) method facilitate the generation of superior design artefacts; what problems are commonly faced by users of the methods; etc. ? Appropriate measures may then be taken to enhance the JSD*(HF) method and support its subsequent introduction. Thus, the uptake of the JSD*(HF) method may be improved.

The above account completes an overall assessment of the JSD*(HF) method.

12.4. Wider Extensions of the Research

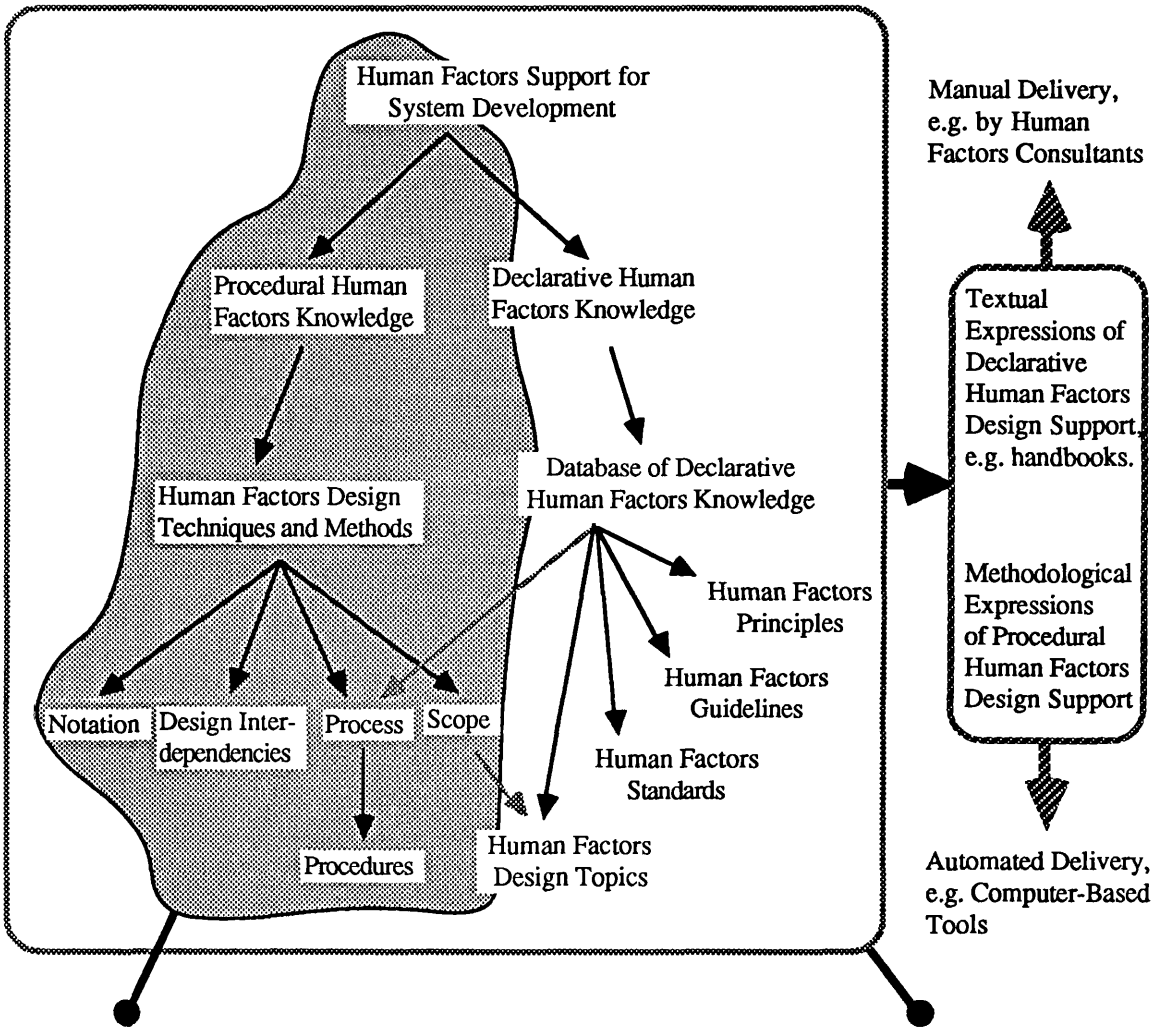
This sub-section is concerned with how the framework of the JSD*(HF) method may be exploited for specifying wider human factors support for system development. To this end, potential extensions of the method may be inferred from Figure 12-1. Specifically, the Figure shows that the design support provided by the JSD*(HF) method is predominantly procedural. Thus, it follows that the scope of the method may be extended to include declarative human factors knowledge. In addition, both types of human factors support may be delivered using computer-based tools. These extensions of the method will now be discussed further.

The incorporation of declarative human factors knowledge into the JSD*(HF) method

The incorporation of declarative human factors knowledge essentially implies an extension of the current user base of the JSD*(HF) method. For instance, the method user base may be extended to include designers who are not human factors experts, e.g. software engineers with some knowledge of human factors (see Figure 12-1, lower left-hand side). To attain this goal, the following objectives would have to be achieved :

(1) explicit links would have to be identified between JSD*(HF) stages and modular human factors design 'topics', e.g. organisational design; socio-

Figure 12-1 : Locating the JSD*(HF) Method within a Range of Human Factors Support for System Development



Human factors design support currently provided by the JSD*(HF) method. General characteristics of the method are as follows :

(a) system design is advanced essentially via separate Human Factors and Software Engineering design streams (attributed partially to the requirement to maintain an unchanged JSD method);
(b) human factors design support is embedded in a collaborative method. In particular, the stage-wise scope, process and notation of the JSD*(HF) and JSD*(SE) methods are linked by specific design inter-dependencies. Thus, context and timing of inter-disciplinary design inputs are defined;
(c) the scope of human factors design support provided by the method is predominantly procedural;
(d) the targeted method user is a human factors designer with a working knowledge of the JSD method.

Both procedural and declarative human factors design support should be provided if method users are not trained specifically in Human Factors.

technical design; job design; training design; personnel selection; workstation design; environment and workplace design; etc. In other words, the design 'topics' have to be set specifically against the stage-wise context defined by the JSD*(HF) method. Thus, appropriate 'repositories' of declarative human factors knowledge may be collated to support each stage of system development as described in (3) below;

(2) design procedures of the JSD*(HF) method would have to be specified at a lower level of description. Thus, a wider user base would be accommodated by the method;

(3) a more comprehensive set of human factors concepts, techniques and reference materials would have to be collated to support method application by a wider range of designers. A prospective set would include the following :

> (a) concepts and techniques for human factors design -- requirements analysis and specification techniques (see Life 1991; Checkland, 1981); socio-technical design techniques (e.g. the Effective Technical and Human Implementation of Computer Systems (ETHICS) method developed by Mumford and Weir, 1979); organisational and participative design techniques (see Eason, 1987; Eason and Cullen, 1988); late evaluation techniques (see Long and Whitefield, 1986);
>
> (b) reference materials for human factors design -- extracts from human factors design handbooks, guidelines, standards and principles, e.g. Smith and Mosier (1984); Helander (1988).

Thus, such an extension of the present research would entail intensive literature surveys to augment each stage of the JSD*(HF) method.

The development of computer-based tools to support the JSD*(HF) and JSD* methods

Three types of computer-based tools may be developed to support the design tasks entailed by the application of the JSD* and JSD*(HF) methods,[9] namely CSCW- (Computer Supported Co-operative Work), IPSE- (Integrated Project Support Environment) and CASE- (Computer Aided Software Engineering) type tools. Presently, the potential for computer-based tool support for each of the methods is discussed.

First, IPSE- and CSCW-type computer-based tools may be developed to facilitate collaborative design management as defined by the JSD* method. For instance, the following computer-based tools may be considered :

(1) CSCW-type tools may be developed to facilitate information exchanges between JSD*(HF) and JSD*(SE) designers, e.g. automatic despatch of information associated with design inter-dependencies of the method. Thus, by despatching design information as soon as it becomes available, such computer-based functions could improve design collaboration;
(2) IPSE-type tools may be developed to support the management of JSD*(HF) and JSD*(SE) design, e.g. overall project planning and tracking of collaborative design advancements. In addition, adherence to the design process of the JSD* method may be enforced by the tool via appropriate inter-locks *across* JSD*(HF) and JSD*(SE) design *streams*. For instance, the tool may be designed to block the documentation of JSD Model specifications (JSD*(SE) stream) until products of the Composite Task Model Stage (JSD*(HF) stream) have been specified. Design inter-dependencies may be supported similarly by the tool.

---

[9] Note that IPSE and CASE tools for the JSD*(SE) method are already available, e.g. MacPDF (Macintosh-based Program Development Facility developed by RARDE); PDF™ (Program Development Facility) and SpeedBuilder™ (both developed by Michael Jackson Systems Limited, now Learmont and Burchett Management Systems Limited). For this reason, the development of computer-based tools for the JSD*(SE) method is excluded from the present account.

Second, computer-based tools for the JSD*(HF) method should provide CASE- and IPSE-type design support. For instance, the following computer-based tools may be considered :

(1) IPSE-type tools may be developed to support the management of JSD*(HF) design, e.g. project planning and tracking to chart the completion of JSD*(HF) deliverables against planned schedules. In addition, adherence to the design process of the JSD*(HF) method may be enforced by the tool via appropriate inter-locks *among* its design *stages*. For instance, the tool may be designed to block the documentation of display designs until a composite task model has been specified. To this end, a simple design may be to implement a 'signing-off' scheme for the designer to indicate the satisfactory completion of each JSD*(HF) product. The tool may then counter-check the input by searching through entry fields designated for the documentation of JSD*(HF) products. If all entry fields have been completed, the tool may then release the inter-locks to permit the documentation of later JSD*(HF) design products. Thus, conceptual task description may be encouraged prior to interaction level specification;
(2) CASE-type tools may be developed to support JSD*(HF) design specification. In particular, the following functional supports should be considered :

(i) text and graphics editors to facilitate design documentation;
(ii) consistency checkers to ensure appropriate application of notational constructs, and the consistent propagation of structured diagram descriptions of products *across* succeeding stages of the JSD*(HF) method;
(iii) simulators to investigate the performance of alternative designs, e.g. MicroSaint™ tools such as MicroSaint Human Operator Simulator (MS HOS)™, and HARDMAN III™ Manpower and Personnel Integration (MANPRINT) toolset (see Dahl et al, 1991a, b);
(iv) prototypers and animators to demonstrate promising designs during feedback elicitation and analysis.

In conclusion, it should be emphasised that the JSD* and JSD*(HF) methods have been developed sufficiently to support the specification of computer-based tools.[10] Pending further extensions of the JSD*(HF) method, computer-based tools may also be developed to support the application of declarative human factors knowledge (see Perlman, 1988).

12.5. Concluding Summary

The primary assertion of the thesis is that current problems of human factors input to system development would be alleviated by integrating Human Factors and Software Engineering methods throughout the system design cycle. It was also argued that methodological integration would be facilitated by the explicitly defined characteristics of SADMs. In particular, the well defined stage-wise scope, process and notation of such methods would facilitate the identification of intersecting Human Factors and Software Engineering design concerns.

To develop these assertions and arguments further, a research plan and a set of strategies for achieving methodological integration were specified. In particular, the plan and strategies address the development and subsequent integration of a structured human factors method with a particular SADM.

Methodological integration was achieved in two sequential specification-and-test phases comprising a number of case-study iterations and design scenarios. Specifically, the sequential phases involve the following :

> (a) the development of a structured human factors method to complement the SADM chosen for integration. Method development was pursued in two modules. The first module addresses the scope and process of the method, while the second was concerned with its notation. Thus, a structured human factors method was developed iteratively;

---

[10] However, the development of CASE-type tools to generate code automatically from JSD*(HF) design descriptions will require tighter specifications of its notational rules.

(b) an explicit specification of design inter-dependencies between the structured human factors method and the SADM. Thus, the methods were integrated and operationalised to support timely and contextually relevant human factors input throughout the system design cycle. In this way, a more effective uptake of human factors contributions may accrue.

The research plan was then instantiated for the JSD method. Thus, human factors design deficiencies of the JSD method were identified and design support requirements to be satisfied by the structured human factors method were defined. On this basis, the JSD*(HF) method was developed and its design stages inter-woven appropriately with the JSD method. Specific design inter-dependencies were also defined between the JSD*(HF) and JSD methods. To account for the design inter-dependencies, the JSD method was re-named the JSD*(SE) method. Throughout method development, case-study assessments were conducted. Following several iterations involving various case-study systems and design scenarios, a promising integrated method, termed the JSD* method, was derived. The method constitutes the primary product of the present research.

To conclude, the foundation established by the present work would support two directions of future research. First, the method has been developed sufficiently to support field trials.[11] In particular, such studies of 'real world' application of the method could uncover information that would contribute significantly to later enhancements of the JSD*(HF) method (and by implication the JSD* method).[12]

---

[11] Such a follow-up project has been commissioned by the sponsors of the present research. The objective of the follow-up project is to test the method further using a larger and more complex case-study system, namely a military command, control and communications system. At a minimum, the project will repeat the case-study tests described in this thesis and address a similar set of methodological issues, e.g. effects of system scale-up; efficacy of its support for design specification; appropriateness of its design inter-dependencies; etc.

[12] As with any design method, the JSD*(HF) and JSD* methods are expected to 'evolve' with its application in the field. For instance, SSADM and JSD have both undergone several updates as part of their maturation process.

Second, the explicitly defined stage-wise scope, process and notation of the JSD*(HF) method could be used to support complementary forms of human factors input, e.g. design principles, guidelines and computer-based tools. In other words, its broad and explicit methodological characteristics constitute a definition of the system design context. Thus, the human factors support required at each stage of system development may be identified. On this basis, appropriate human factors principles, guidelines and computer-based tools may be recruited or developed to support the system design cycle. These concerns may be pursued further in a follow-up research project.

PART VI :

References

Akscyn, R. M., and McCracken, D. L., (1984), ZOG and the USS CARL VINCENT : Lessons in System Development. In : B. Shackel (ed.), Proceedings of the First IFIP Conference on Human-Computer Interaction (Interact '84), Vol. 2, pp. 303-308, London, Elsevier Science Publishers, North-Holland.

Alexander, H., (1987), Executable Specifications as an Aid to Dialogue Design. In: H. J. Bullinger and B. Shackel (eds.), Proceedings of the Second IFIP Conference on Human-Computer Interaction (Interact '87), pp. 739-744, London, Elsevier Science Publishers, North-Holland.

Alvey Human Interface Committee Report, (1987), Human Interface Issues in the IT'86 Programme, pg. 15, 27, 32, 33 and 34, July 1987.

Alvey Man-Machine Interface Workshop, (1984), User Interface Design and Design Methods, Coventry, pg. 20, 27, 28, 37, 41 and 89, February 1984.

Alvey Man-Machine Interface Project 143, (1988), User Skill Task Match Methodology, Final Report, April 1988.

Anderson, J. M., (1988), The Integration of HCI Principles in Structured System Design Principles. In : Proceedings of Milcomp'88, Military Computers, Graphics and Software, September 1988, London.

Annett, J. and Duncan, K. D., (1967), Task Analysis and Training Design. Occupational Psychology, 41, pp. 211-221.

Benbasat, I. and Wand, I. (1984), A Structured Approach to Designing Human-Computer Dialogues, In : International Journal of Man-Machine Studies, Vol. 21, pp 105-126, Academic Press Inc., London.

Benyon, D. and Skidmore, S., (1987), Towards a Tool Kit for the Systems Analyst, The Computer Journal, Vol. 30, No. 1, 1987, pp. 2-7, Cambridge University Press.

Berns, T. A. R., (1984), The Integration of Ergonomics into Design, Behaviour and Information Technology, Vol. 3, pp. 277-284, London : Taylor Francis.

Birchenough, A. and Cameron, J. R., (1989), JSD and Object-Oriented Design, In : J. R. Cameron, JSP and JSD : The Jackson Approach to Software Development, Second Edition, pp. 293-304, IEEE Computer Society Press.

Blyth, R. C. and Hakiel, S. R., (1988), A Methodology for Man-Machine Interface Design, Internal Plessey Research Report 72/88/R467C, Roke Manor.

Blyth, R. C. and Hakiel, S. R., (1989), A User Interface Design Methodology and the Implication for Structured Systems Design Methods. In : Proceedings of the IEE Third International Conference on Command, Control, Communications and Management Information Systems, Bournemouth, UK, 2-4 May.

Boar, B. H., (1984), Application Prototyping : A Requirements Definition Strategy for the 80's, New York : John Wiley and Sons Inc., 1984.

Bóhm, B. W., (1976), Software Engineering, IEEE Transactions on Computing, December 1976, pp. 1226-1241.

Bóhm, B. W., (1981), Software Engineering Economics, Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

Bóhm, B. W., (1984), Software Life-Cycle Factors, In : C. R. Vick and C. V. Ramamoorthy (eds.), Handbook of Software Engineering, New York : van Nostrand Reinhold Co., 1984.

Boldyreff, C., (1986), Using Jackson Structured Programming to Describe Syntax, Computing Notebook Software, March 26 1986.

Borufka, H. G., Kuhlmann, H. W. and ten Hagen, P. J. W., (1982), Dialogue Cells : A Method for Defining Interactions, In : IEEE Computer Graphics and Applications, July, pp 25-33.

Bott, M. F., (1988), Systems Analysis and Design Methods and Integrated Development Environments : The State of the Art and the Future. In E. D. Megaw (ed.), Contemporary Ergonomics 1988, 'Ergonomics Giving Quality to Life', Proceedings of the Ergonomics Society 1988 Annual Conference, pp. 164-170, London: Taylor Francis.

BS 6719: 1986, Specifying User Requirements for a Computer-Based System, British Standards Institute.

Bury, K. F., (1984), The Iterative Development of Usable Computer Interfaces. In : B. Shackel (ed.), Proceedings of the First IFIP Conference on Human-Computer Interaction (Interact '84), Vol. 2, pp. 343-348, London, Elsevier Science Publishers, North-Holland.

Butler, K., Bennett, J., Polson, P., and Karat, J., (1989), Report on the Workshop on Analytical Models, SIGCHI Bulletin, Vol. 20, No. 4, April 1989, pp. 63-77.

Buxton, W., (1983), Lexical Considerations of Input Structures, In : ACM SIGGRAPH Computer Graphics, 17, pp 31-37.

Buxton, W., Lamb, M. R., Sherman, D. and Smith, K. C., (1983), Towards a Comprehensive UIMS, Computer Graphics, Vol. 17, No. 3, July 1983, pp. 35-42.

Cameron, J. R., (1987), Mapping JSD Network Specifications into ADA, ADA User, Vol. 8, Supplement, pp. 591-599, ADA Language U.K. Ltd.

Cameron, J. R., (1989), JSP and JSD : The Jackson Approach to Software Development, Second Edition, IEEE Computer Society Press.

Card, S. K., Moran, T. P. & Newell, A., (1983), The Psychology of Human-Computer Interaction, Hillsdale, N.J. : Lawrence Erlbaum Associates.

Carroll, J. M. and Campbell, R. L., (1986), Softening Up Hard Science : Reply to Newell and Card, Human Computer Interaction, 1986, Vol. 2, No. 3, pp. 227-250, Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey.

Carver, M. K., (1988), Practical Experience of Specifying the Human-Computer Interface Using JSD. In : E. D. Megaw (ed.), Contemporary Ergonomics, Proceedings of the Ergonomics Society's 1988 Annual Conference, Manchester, pp. 177-182, London : Taylor Francis.

Carver, M. K. and Cameron, J., (1987), The Jackson System Development Method : A Framework for the Specification of the Human-Computer Interface. Unpublished Internal Report of Michael Jackson Limited, 1987.

Carver, M. K., Clenshaw, D. C., Myles, D. J. L. and P. J. Barber, (1987), Final Report on the Use of JSD in the Design of an ADP System to Provide Support to the Corps All Sources Cell, Internal RARDE Report.

CCTA (Draft) Report, (1988), User System Interaction Strategy Scoping Study. CCTA Office Systems Branch, June 1988.

Chapanis, A., (1965), On the Allocation of Functions between Men and Machines, Occupational Psychology, 39, pp. 1-11, 1965.

Chapanis, A. and Budurka, W. J., (1990), Specifying Human-Computer Interface Requirements, Behaviour and Information Technology, Vol. 9, No. 6, pp. 479-492, London : Taylor Francis.

Checkland, P., (1981), Systems Thinking and Systems Practice, Chichester, John Wiley Publishers.

Chikofsky, E. J. and Cross II, J. H., (1990), Reverse Engineering and Design Recovery : A Taxonomy, IEEE Software, Vol. 7, No. 7, pp. 13-17, January 1990, IEEE Computer Society Press.

Clark, I. A., and Howard, S., (1988), Facading a Medical Application for Usability and Utility. In : Proceedings of the APL Conference, University of Kent, September 1988.

Clegg, C., Ravden, S., Corbett, M. and Johnson, G., (1989), Allocating Functions in Computer Integrated Manufacturing : A Review and a New Method, Behaviour and Information Technology, 1989, Vol. 8, No. 3, pp. 175-190, London : Taylor Francis.

Clowes, I., (1986), Methodology for Designing Adaptive User Interfaces, Alvey MMI/006 AID Project Final Report aid/2.41/concept/s0111.3.

Clowes, I., (1988), Models of Users in System Design, Alvey User Modelling SIG Workshop Handout, Cosner's House, 6-7 June 1988.

Coles, S. and Lim, K. Y., (1989), Towards a Task Analysis Method for JSD*: The Usefulness of Generification Procedures, RARDE Project Internal Working Document Number 30.

Coutaz, J., (1985), Abstractions for User Interface Design, In : IEEE Computer, 18, 21-34.

Coutaz, J., (1989), UIMS : Promises, Failures and Trends, In : People and Computer IV, Sutcliffe, A. and Macaulay, L. (eds.), Proceedings of the Fifth Conference of the BCS HCI SIG (HCI'89), pp. 71-84, Nottingham, Cambridge University Press.

Coyne, R., (1988), Logic Models of Design, Pitman Publishing London.

Crinnion, J., (1989), A Role for Prototyping in Information Systems Design Methodology, Design Studies, Vol. 10, No. 3, July 1989, pp. 144-150, Butterworth Scientific Ltd, London.

Dahl, S., Laughery, K. R. and Hood, L., (1991a), Integrating Task Network and Anthropometric Models, In ed. E. Lovesey, Proceedings of the Ergonomics Society's 1991 Annual Conference, pp. 151-156, London: Taylor Francis.

Dahl, S., Laughery, K. R. and Hood, L., (1991b), HARDMAN III MANPRINT Tools, Special Workshop at the Ergonomics Society's 1991 Annual Conference, Southampton, April 1991.

Damodaran, L., (1988), DIADEM, Ergonomics Society Workshop on SSADM and Task Analysis, May 1988.

Damodaran, L., Ip, K. and Beck, M., (1988), Integrating Human Factors Principles into Structured Design Methodology : A Case-Study in the UK Civil Service. In : H. J. Bullinger et al (eds.), Information Technology for Organisational Systems, Elsevier Science Publishers, pp. 235-241, North Holland.

Diaper, D., (1989a), Task Analysis for Human-Computer Interaction, Ellis Horwood Books in Information Technology, John Wiley and Sons.

Diaper, D., (1989b), Knowledge Elicitation : Principles, Techniques and Applications, Ellis Horwood Books in Expert Systems, John Wiley and Sons.

Dowell, J., and Long, J. B., (1989), Towards a Conception for an Engineering Discipline of Human Factors. In : P. Barber and J. Laws (eds.), Ergonomics (Special Issue) on "Methodological Issues in Cognitive Ergonomics", Vol. 32, No. 11, pp. 1513-1536, London: Taylor Francis.

Drury, C. G., (1983), Task Analysis Methods in Industry, In : Applied Ergonomics, 14, 1, pp. 19-28.

Duncan, K., (1974), Analysis Techniques in Training Design, In Edwards, B. and Lees, F. P., The Human Operator in Process Control, London: Taylor and Francis.

Eason, K. D., (1987), Methods of Planning the Electronic Workplace, Behaviour and Information Technology, Vol. 6, No. 3, pp. 229-238, London : Taylor Francis.

Eason, K. D., (1988), Information Technology and Organizational Change, London : Taylor & Francis.

Eason, K. D. and Cullen, J., (1988), Human Factors Contributions in the Context of I.T. System Design and Implementation. In : H. J. Bullinger et al (eds.), Information Technology for Organisational Systems, pp. 811-816, Elsevier Science Publishers, North-Holland.

Esgate, A., Whitefield, A. and Life, A., (1990), Developing Usability Integration Principles for the Design of IBCN Systems, In : G. van der Veer, et al (eds.), Proceedings of the Fifth European Conference on Cognitive Ergonomics, Urbino (Italy), September 1990, pp. 359-374, Golem Press.

ESPRIT 385, (1989), In : E. D. Megaw (ed.), Contemporary Ergonomics, Proceedings of the Ergonomics Society's 1989 Annual Conference, Reading, pp. 82-131, London : Taylor Francis.

ESPRIT 385, (1990), In : D. Diaper et al (eds.), Proceedings of the Third IFIP Conference on Human-Computer Interaction (Interact '90), pp. 371-382, Cambridge, Elsevier Science Publishers, North-Holland.

Essink, L. J. B., (1988), A Conceptual Framework fro Information Systems Development Methodologies. In : H. J. Bullinger et al (eds.), Information Technology for Organisational Systems, pp. 354-362, Elsevier Science Publishers, North-Holland.

Fáhnrich, K., Fauser, A. and Heller, N., (1984), The Extent of Introduction of Electronic Machinery in the Office. Consolidated Report, Dublin : European Foundation for the Improvement of Living and Working Conditions, 1984.

Farooq, M. U. and Dominick, D., (1988), A Survey of Formal Tools And Models For Developing User Interfaces, In : International Journal of Man-Machine Studies Vol. 29, pp. 479-496, Academic Press Inc., London.

Finkelstein L. and Finkelstein, A., (1983), Design Aids, In : Fehrenbach, P. (ed.), Towards a Prototype Interface Design Tool, Alvey Project MMI/142, Final Report.

Finkelstein, A. and Potts, C., (1985), Evaluation of Existing Requirements Extraction Strategies, FOREST Report R1.

Fitter, M. and Green, T. R. G., (1979), When Do Diagrams Make Good Computer Languages ? International Journal of Man-Machine Studies, Vol. 11, pp. 235-261, 1979, Academic Press Inc., London.

Fitts, P. M., (1962), Functions of Men in Complex Systems, Aerospace Engineering, Vol. 21 (1), pp. 34-39, 1962.

Fitzgerald, G., (1988), Information Systems Development for Changing Environments : Flexibility Analysis. In : H. J. Bullinger et al (eds.), Information Technology for Organisational Systems, pp. 587-592, Elsevier Science Publishers, North-Holland.

Fleishman, E. A. and Quaintance, M. K., (1984), Taxonomies of Human. Performance, Academic Press Inc., 1984.

Foley, J. D. and van Dam, A., (1982), Fundamentals of Interactive Computer Graphics. Englewood Cliffs, NJ : Prentice-Hall.

Foley, J. D. and Wallace, V. L., (1974), The Art of Natural Graphic Man-Machine Conversation, In : Proceedings of the IEEE, 63, pp 462-471.

Foley, J. D., Wallace, V. L. and Chan, P., (1984), The Human Factors of Computer Graphics Techniques, In : IEEE Computer Graphics and Applications, 4, pp. 13-47.

Fox, J. M., (1982), Software and its Development. Englewood Cliffs, New Jersey: Prentice-Hall Inc., 1982.

Frohlich, D. M. and Luff, P., (1989), Some Lessons From an Exercise in Specification, Human-Computer Interaction, 1989, Vol. 4, pp. 121-147, Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey.

Galliers, R. D., (1984), An Approach to Information Needs Analysis. In : B. Shackel (ed.), Proceedings of the First IFIP Conference on Human-Computer Interaction (Interact '84), Vol. 1, pp. 409-418, London, Elsevier Science Publishers, North-Holland.

Gillett, P. E. and Northam, D. J., (1990), Matching Warships and Sailors. In : Proceedings of the Symposium on Human Factors in Warships and Naval Systems, Westminster, November 1990, pp. 84-98.

Glasson, B. C., (1984), Guidelines for User Participation in the System Development Process. In : B.Shackel (ed.), Proceedings of the First IFIP Conference on Human-Computer Interaction (Interact '84), Vol. 1, pp. 284-298, London, Elsevier Science Publishers, North-Holland.

Gould, J. D., (1988), How to Design Usable Systems. In : M. Helander (ed.), Handbook of Human Computer Interaction, pp. 757-790, Elsevier Science Publishers, North Holland, 1988.

Gould, J. D., and Lewis, C., (1983), Designing for Usability -- Key Principles and What Designers Think. In : Proceedings of CHI'83 Human Factors in Computing Systems, Boston, December 1983, pp. 50-53, ACM, New York.

Grandjean, E., (1984), Forward, Ergodesign '84, Behaviour and Information Technology, Vol. 3, pp. 261-262, London : Taylor Francis.

Grandjean, E., (1988), Fitting the Task to the Man : An Ergonomic Approach, 4th edition. London : Taylor & Francis.

Grønbák, K., (1989), Rapid Prototyping with Fourth Generation Systems -- An Empirical Study, Office : Technology and People, 5:2, 1989, pp. 105-125, Elsevier Science Publishers Ltd, England.

Grudin, J., Ehrlich, S. F. and Shriner, R., (1987), Positioning Human Factors in the User Interface Development Chain. In : CHI + GI 1987, pp. 125-131, ACM Press.

Guest, S. P., (1982), Software Tools for Dialogue Design, In : International Journal of Man-Machine Studies, Vol. 14, pp. 263-385, Academic Press Inc., London.

Guest, S., (1988), Human Factors in Telematic Systems, Alvey HI Club, User Modelling SIG Workshop Handout on "User's Models of Systems", 29 March'88.

Hakiel, S. R. and Blyth, R. C., (1990a), Keeping the Human in Context, In : E. Lovesey (ed.), Contemporary Ergonomics 1990, 'Ergonomics Setting Standards for the '90s', Proceedings of the Ergonomics Society's 1990 Annual Conference, Leeds, 3-6 April, pp. 123-128, Taylor and Francis.

Hakiel, S. R. and Blyth, R. C., (1990b), Keeping the Human in Context, Contemporary Ergonomics 1990, 'Ergonomics Setting Standards for the '90s', Presentation OHPs, Leeds, 3-6 April.

Hammond, N., Jorgensen, A., MacLean, A., Barnard, P. and Long, J. B. (1983), Design Practice and Interface Usability : Evidence from Interviews with Designers, In : Proceedings of the CHI'83 Conference, pp. 40-44, ACM Press.

Hares, J., (1987), Methods for a Longer Life, Computer News/Databases, pg. 18, 6 August, 1987.

Hartson, H. R. and Hix, D., (1988), Human-Computer Interface Development : Concepts and Systems for its Management. In Proceedings of Tutorial Sessions, CHI '88, New York: ACM Press.

Hartson, H. R. and Hix, D., (1989), Towards Empirically Derived Methodologies and Tools for Human-Computer Interface Development, International Journal of Man-Machine Studies, Vol. 31, pp. 477-494, Academic Press.

Haubner, P. J., (1990), Ergonomics in Industrial Product Design, Ergonomics, 1990, Vol. 33, No. 4, 477-485, London : Taylor Francis.

Hekmatpour, S. and Ince, D. C., (1987), Evolutionary Prototyping and the Human Computer Interface, In : H. J. Bullinger and B. Shackel (eds.), Proceedings of the Second IFIP Conference on Human-Computer Interaction (Interact '87), pp. 479-484, London, Elsevier Science Publishers, North-Holland.

Helander, M., (1988), Handbook of Human Computer Interaction, Elsevier Science Publishers, North Holland.

Hewett, J. and Durham, T., (1987), Computer-Aided Software Engineering: Commercial Strategies, Ovum Ltd.

Hirsch, R. S., (1984), VDTs and the Human Factors Community: Tipping the Iceberg, Human Factors Society Bulletin, Vol. 27, No. 6, pp. 1-3.

Hirschheim, R., (1985), Office Automation : A Social and Organisational Perspective. Chichester : Wiley, 1985.

Hutchins, E., (1987), Metaphors for Interface Design, Internal Technical Report, Institute of Cognitive Science, University of California, San Diego.

Ip, W. K., Damodaran, L., Olphert, C. W. and Maguire, M. C., (1990), The Use of Task Allocation Charts in System Design : A Critical Appraisal. In : D. Diaper et al (eds.), Proceedings of the Third IFIP Conference on Human-Computer Interaction (INTERACT '90), pp. 289-294, Elsevier Science Publishers, North-Holland.

Innocent, P. R., (1982), Towards Self-Adaptive Interface Systems, International Journal of Man-Machine Studies, 16(3), April 1982, pp. 287-299, Academic Press Inc., London.

ISO/TC/159/SC4/WG5 N84, Criteria for the Evaluation of Software -- Software Quality Characteristics. ISO Standards Working Papers.

Jackson, M. A., (1983), System Development, Englewood Cliffs New Jersey : Prentice-Hall International.

Jensen, R. W. and Tonies, C. C. (eds.), (1979), Software Engineering, Englewood Cliffs, New Jersey : Prentice-Hall Inc., 1979.

Johnson, P, (In Press), Models in Human Computer Interaction, In : Human Factors for Interactive Systems, Chapter Five.

Johnson, P., Diaper, D. and Long, J. B., (1984), Tasks, Skill and Knowledge; Task Analysis for Knowledge Based Descriptions. In : B. Shackel (ed.), Proceedings of the First IFIP Conference on Human-Computer Interaction (Interact '84), Vol. 1, pp. 23-28, London, Elsevier Science Publishers, North-Holland.

Johnson, P. and Johnson, H., (1987), Generification : A Process of Identifying Generic Properties of Tasks within a given Domain. Queen Mary College Report to ICL, No. 2, 1987.

Johnson, P. and Johnson, H., (1988), Knowledge Analysis of Tasks : Theory, Method, and Suggestions for Application to System Design. Internal Technical Report, Queen Mary College, University of London.

Johnson, P., Johnson, H. and Russell, F., (1988), Collecting and Generalising Knowledge Descriptions from Task Analysis Data, ICL Technical Journal.

Jones, J. C., (1973), Design Methods : Seeds of Human Futures, Wiley Inter-Science, pp. 123-133, John Wiley and Sons Ltd., London.

Keane, M. and Johnson, P., (1987), Preliminary Analysis for Design, In : Diaper, D. and Winder, R. (eds)., Proceedings of the BCS HCI SG Conference, HCI'87 - People and Computers III, pp. 133-146, Exeter 7-11 September, Cambridge University Press.

Keller, R., (1987), Expert System Technology, Yourdon Press, Englewood Cliffs, New Jersey, 1987.

Kieras, D. and Polson, P. G., (1985), An Approach to the Formal Analysis of User Complexity, In : International Journal of Man-Machine Studies, Vol. 22, pp. 365-394, Academic Press Inc., London.

Klein, G. A. and Brezovic, (1986), Design Engineers and the Design Process : Decision Strategies and Human Factors Literature. In : Proceedings of the Human Factors Society -- 30th Annual Meeting, 1986, pp. 771-775.

Klein, L. and Newman, W., (1987), Quality Assurance Aspects of IT'86. In : Alvey Human Interface Club, Report of Open Meeting, Strand, London, 13 October 1987, pp. 41-66.

Kloster, G. V. and Tischer, K., (1984), Man-Machine Interface Design Process. In : B. Shackel (ed.), Proceedings of the First IFIP Conference on Human-Computer Interaction (Interact '84), Vol. 2, pp. 236-241, London, Elsevier Science Publishers, North-Holland.

Life, A., (1991), A Structured Analysis and Design Method for User Requirements Specification, Ergonomics Unit, University College London, JCI Research Proposal.

Lim, K. Y., (1986), Display Structure and Memory Location Task Performance. MSc (Ergonomics) Dissertation, University of London.

Lim, K. Y., (1987), An Assessment of the Macdraw Application Package, RARDE Project Internal Working Document Number 9.

Lim, K. Y., (1988a), Task Strategies, Metaphors and the Integration of Interface Design into JSD, RARDE Project Internal Working Document Number 15.

Lim, K. Y., (1988b), Essential Considerations for Interfacing Task Analysis and Task Modelling with JSD Modelling, RARDE Project Internal Working Document Number 17.

Lim, K. Y., (1988c), On Reasoning about the Project and DDN Case-Study Proposals, RARDE Project Internal Working Document Number 18.

Lim, K. Y., (1988d), On Reasoning about User Interface Design using Extended JSD in Conjunction with an Extant Systems System Analysis Approach, RARDE Project Internal Working Document Number 19.

Lim, K. Y., (1988e), Some Considerations on Notational Requirements for Task and Interface Information Capture : Towards the Conception of a JSD* Notation, RARDE Project Internal Working Document Number 23.

Lim, K. Y., (1989a), A Perspective on HCI Model Classes and their Life-Cycle with respect to the System Design Process : Providing a Rational Basis for the Current Conception of JSD*, RARDE Project Internal Working Document Number 26.

Lim, K. Y., (1989b), Case-Study Illustrations of a Conception of Structured Interface Design within JSD*, RARDE Project Internal Working Document Number 27.

Lim, K. Y., (1989c), Towards a JSD* Method -- A Review of the Research Scope, Requirements and Constraints; and the Proceduralisation of the JSD* User Interface Design Process, RARDE Project Internal Working Document Number 28.

Lim, K.Y., (1989d), A Preliminary Conception of the Stage-wise Process of User Interface Design Within JSD* : A Case-Study Instantiation Using the University College London Recreational Facility Booking System (UCL RFBS), RARDE Project Internal Working Document Number 29.

Lim, K. Y., (1990a), An Overview of the First Pass Version of the JSD*(HF) Method, RARDE Project Internal Working Document Number 33.

Lim, K. Y., (1990b), JSD*(HF) Deliverables Corresponding to the First Conjunction with the JSD*(SE) Design Stream -- Outputs of the CTM and SUN Stages for the Trouble-Shooting Module of the DDN NMS Case-Study, RARDE Project Internal Working Document Number 37A.

Lim, K. Y., (1990c), The DDN NMS Case-Study -- Documenting Outputs of the ESSA and GTM Stages for the Trouble-Shooting Module and Associated Refinements Suggested for the JSD*(HF) Method, RARDE Project Internal Working Document Number 38A.

Lim, K. Y., (1990d), The DDN NMS Case-Study -- Documenting Outputs of the CTM and Post-CTM Stages for the Trouble-Shooting Module and Associated Refinements Suggested for the JSD*(HF) Method, RARDE Project Internal Working Document Number 39A.

Lim, K. Y., (1990e), JSD*(HF) Specifications of the User Interface for the DDN NMS Case-Study (Trouble-Shooting Module), RARDE Project Internal Working Document Number 40A.

Lim, K. Y., (1991), An Energy Management System for the Home : Human Factors Evaluation of the SmallTalk Prototype, London HCI Centre Report, LHC/9013/REP1, January 1991.

Lim, K. Y., Long, J. B. and Silcock, N., (1992), Integrating Human Factors with the Jackson System Development Method : An Illustrated Overview, In : Barber, P. and Laws, J. (eds.), Ergonomics Special Issue on Methodological Issues in Cognitive Ergonomics III, Vol 34, Taylor & Francis London.

Long, J. A. and Neale, I. M., (1989), Validating and Testing in KBS -- A Case-Study. In : Proceedings of the Second International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Tennessee, U.S.A., June 1989.

Long, J. B., (1986), Use and Abuse of Models : HI and IKBS Perspectives, In : Conference Record of the Alvey Conference, University of Sussex, 1-4 July 1986, pp. 40-41.

Long, J. B., (1987), A Framework for User Models, In : E. D. Megaw (ed.), Contemporary Ergonomics - Proceedings of the Ergonomics Society 1987 Annual Conference, pp. 245-250, Swansea, 6-10 April, Taylor & Francis.

Long, J. B. and Dowell, J., (1989), Conceptions of the Discipline of HCI : Craft, Applied Science, and Engineering. In : A. Sutcliffe and L. Macaulay (eds.), People and Computers V, Proceedings of the Fifth Conference of the British Computer Society Human-Computer Interaction Specialist Group, Nottingham, September 1989, pp. 9-34, Cambridge University Press.

Long, J. B. and Whitefield, A. D., (1986), Evaluating Interactive Systems, HCI'86 Tutorial, York, September 1986.

Lucas, H. C., (1975), Why Information Systems Fail, Columbia University Press, New York, 1975.

Lundell, J. and Notess, M., (1991), Human Factors in Software Development : Models, Techniques, and Outcomes, In : P. Robertson, G. M. Olson, and J. S. Olson (eds.), Proceedings of the CHI'91 Conference, New Orleans, May 1991, pp. 145-152, ACM Press, New York.

Maddison, R. N., (1983), Information System Methodologies, In : P. A. Samet (ed.), BCS Monographs in Informatics, Wiley Heyden.

Maguire, M., (1982), An Evaluation of Published Recommendations on the Design of Man-Computer Dialogues, International Journal of Man-Machine Studies, Vol. 16, No. 3, pp. 237-261, Academic Press Inc., London.

Mantei, M., (1986), Techniques for Incorporating Human Factors in the Software Life-Cycle. In : Proceedings of STA-III Conference : Structured Techniques in the Eighties : Practice and Prospect, Chicago, June 1986.

Mantei, M., and Teorey, T. J., (1988), Cost/Benefit Analysis for Incorporating Human Factors in the Software Life-Cycle, Computing Practices, Communications of the ACM, Vol. 31, No. 4, pp. 428-439, April 1988.

Marshall, C. R., (1984), System ABC : A Case-Study in the Design and Evaluation of a Human-Computer Dialog. In : B. Shackel (ed.), Proceedings of the First IFIP Conference on Human-Computer Interaction (Interact '84), Vol. 1, pp. 419-423, London, Elsevier Science Publishers, North-Holland.

McClleland, I., (1990), Marketing Ergonomics to Industrial Engineers, Ergonomics, 1990, Vol. 33, No. 4, pp. 391-398, London : Taylor Francis.

McKeen, J. D., (1983), Successful Development Strategies for Business Application Systems, Management Information Systems Quarterly, September 1983, pp. 47-65.

McKenzie, J., (1988), Guidelines and Principles of Interface Design. In : N. Heaton and M. Sinclair (eds.), State of the Art Report 15:8, "Designing End-User Interfaces", pp. 73-84, England: Pergamon Infotech Limited.

McNeile, A. T., (1986), Jackson System Development. In : T. W. Olle et al (eds.), Information Systems Design Methodologies : Improving the Practice, pp. 225-246, Elsevier Science Publishers, North Holland.

Meister, D., (1984), A Catalogue of Ergonomic Design Methods. In Proceedings of the International Conference on Occupational Ergonomics, pp. 17-25.

Moraal, J. and Kragt, H., (1990), Macro-Ergonomic Design : The Need for Empirical Research Evidence, Ergonomics, 1990, Vol. 33, No. 5, pp. 605-612, London : Taylor Francis.

Moran T. P. (1981), The Command Language Grammar : A Representation for the User Interface of Interactive Computer Systems, In : International Journal of Man-Machine Studies, Vol. 15, pp. 3-50, Academic Press Inc., London.

Morrison, W., (1988), Communicating with Users during Systems Development, Information and Software Technology, June 1988, Vol. 30, No. 5, pp. 295-298, Butterworth Scientific Ltd, London.

Multi-User Computing, (1989), Automated Tools Become a Reality, pp. 20-26 & pg. 29, January 1989.

Mumford, E. and Weir, M., (1979), Computer Systems in Work Design -- the ETHICS (Effective Technical and Human Implementation of Computer Systems), Associated Business Press London.

Nielsen, J., (1987), The Spectrum of Models in Software Ergonomics, In : Proceedings of the Fifth Symposium on Empirical Foundations of Information and Software Sciences, 23-25 November 1987, Denmark.

Newman, W., (1988), User Interface Design Practice. HCI'88 Conference, Tutorial Notes, Manchester.

Norman, D. A., (1983), Some Observations on Mental Models, In : In Stevens, A. L. and Gentner, D. (eds.), Mental Models, Lawrence Erlbaum Associates Publishers, Hillsdale, New Jersey.

Norman, D. A., (1986), Cognitive Engineering, In : Norman, D. A. and Draper, S. W. (eds.), User Centered System Design : New Perspectives on Human Computer Interaction, pp. 31-61, Lawrence Erlbaum Associates Publishers, Hillsdale, New Jersey.

Norman, M. A., (1988), Developments in Computing and the User Interface -- Emerging Issues in End-User Interface Design. In : N. Heaton and M. Sinclair (eds.), State of the Art Report 15:8, "Designing End-User Interfaces", pp. 85-96, England: Pergamon Infotech Limited.

Norris, M. T., (1985), The Application of Formal Methods in Systems Design, British Telecom Technology Journal, Vol. 3, No. 4, October 1985, pp. 53-59.

O'Niel, D., (1980), The Management of Software Engineering Part II : Software Engineering Program, IBM Systems J., Vol. 19, No. 4, pp. 421-431.

Olle, T. W., (1988), An Information Systems Life-Cycle and Related Concepts. In: H. J. Bullinger et al (eds.), Information Technology for Organisational Systems, pp. 553-559, Elsevier Science Publishers, North-Holland.

Payne, S. J. and Green, T. R. G., (1986), TAG : A Model of the Mental Representation of Task Languages, Human Computer Interaction, 1986, Vol. 2, pp. 93-133, Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey.

Pelly, R. C. and Crampin, T., (1990), Human Factors Today and Tomorrow in Ship Control Centers. In : Proceedings of the Symposium on Human Factors in Warships and Naval Systems, Westminster, November 1990, pp. 99-114.

Perlman, G., (1987), An Overview of SAM : A HyperText Interface to Smith and Mosier's Guidelines for Designing User Interface Software. Tyngsboro, MA: Wang Institute, Wang Institute Technical Report TR-87-09, 1987.

Perlman, G., (1988), Software Tools for User Interface Development. In : M. Helander (ed.), Handbook of Human Computer Interaction, pp. 819-834, Elsevier Science Publishers, North Holland, 1988.

Pikaar, R. N., Lenior, T. M. J. and Rijnsdorp, J. E., (1990), Implementation of Ergonomics in Design Practice : Outline of an Approach and some Discussion Points, Ergonomics, 1990, Vol. 33, No. 5, 583-587, London : Taylor Francis.

Preece, J., Woodman, M., Ince, D. C., Griffiths, R. and Davies, G., (1987), Towards a Structured Approach to Specifying User Interface Design, In : H. J. Bullinger and B. Shackel (eds.), Proceedings of the Second IFIP Conference on Human-Computer Interaction (Interact '87), pp. 415-421, London, Elsevier Science Publishers, North-Holland.

Price, H. E., (1985), The Allocation of Functions in Systems, Human Factors, Vol. 27, No. 1, pp. 33-45, Human Factors Society Inc., Santa Monica, U.S.A.

Poltrock, S. E., Steiner, D. D. and Tarlton, P. N., (1986), Graphics Interfaces for Knowledge-Based System Development, In : Proceedings of CHI'86 Conference : Human Factors in Computing Systems, pp. 9-15, ACM: New York.

Ragoczei, S. and Hirst, G., (1990), The Meaning Triangle as a Tool for the Acquisition of Abstract, Conceptual Knowledge, International Journal of Man-Machine Studies, Vol. 33, pp. 505-520, Academic Press Ltd.

Rasmussen, J., (1986), Information Processing and Human-Computer Interaction : An Approach to Cognitive Engineering, Elsevier Science North Holland.

Reisner, P., (1977), Use of Psychological Experimentation as an Aid to the Development of a Query Language, In : IEEE Trans. in Software Engineering, SE-3, pp. 218-229.

Rekoff Jr., M. G., (1985), On Reverse Engineering, IEEE Trans. Systems, Man, and Cybernetics, pp. 244-252, March-April 1985.

Renold, A., (1989), Jackson System Development for Real Time Systems, In : J. R. Cameron, JSP and JSD : The Jackson Approach to Software Development, Second Edition, pp. 235-278, IEEE Computer Society Press.

Robertson, D., (1987), Closing the Gap. In : Alvey Human Interface Club, Report of Open Meeting, Strand, London, 13 October 1987, pp. 67-74.

Rogers, J. G. and Pegden, C. D., (1977), Formatting and Organisation of a Human Engineering Standard, Human Factors, Vol. 19, No. 1, pp. 55-61.

Rosson, M. B., (1987), Real World Design, SIGCHI Bulletin, Vol. 19, No. 2, October 1987, pp. 61-62.

Rouse, W. B. and Boff, K. R., (eds., 1987), System Design : Behavioural Perspectives on Designers, Tools and Organisations, Elsevier Science North Holland.

Rubinstein, R. and Hersh, H., (1984), The Human Factor : Designing Computer Systems for People, Digital Press, Bedford, Massachusetts, 1984.

Sebillotte, S., (1988), Hierarchical Planning as a Method for Task Analysis: The Example of Office Task Analysis, In : Behaviour and Information Technology, Vol. 7, No. 3, 275-293, London : Taylor Francis.

Selby, C. and Long, J. B., (1991), Investigating the Ease of Use of Object-Oriented Interfaces : Apple LISA™ vs IBM TopView™, In : E. Lovesey (ed.), Contemporary Ergonomics 1991, 'Ergonomics -- Design for Performance, Proceedings of the Ergonomics Society's 1991 Annual Conference, Southampton, 16-19 April, pp. 175-182, Taylor and Francis.

Shackel, B., (1985), Ergonomics in Information Technology in Europe -- A Review, Behaviour and Information Technology, Vol. 4, No. 4, pp. 263-287, London : Taylor Francis.

Shackel, B., (1986a), Ergonomics in Design for Usability. In : M. D. Harrison and A. F. Monk (eds.), People and Computers : Designing for Usability, pp. 44-64, 1986, Cambridge University Press.

Shackel, B., (1986b), IBM Makes Usability as Important as Functionality, The Computer Journal, Vol. 29, pp. 475-476.

Sharratt, B. D., (1987), Top Down Interactive Systems Design : Some Lessons Learnt from Using Command Language Grammar, In : Bullinger, H. J. and Shackel, B. (eds.), Proceedings of the INTERACT '87 Conference on Human-Computer Interaction, pp. 395-399, IFIP, Elsevier Science, Holland.

Sharratt, B. D., (1988), Is CLG Usable ? SIGCHI Bulletin, April 1988, Vol. 19, No. 4, pp. 38-39.

Shneiderman, B., (1987), Designing the User Interface : Strategies for Effective Human-Computer Interaction, Addison-Wesley Publishing Company.

Shuttleworth, M., (1987), The Role of Application Developers, Graphics Systems Centre, ICL Office Systems, Human Computer Co-Operation, HCC/1/15 (Issue 1), March 1987.

Silcock, N., (1989), Towards a Task Analysis Method for JSD* : An Analytical and Procedural Review of "Hierarchical Task Analysis"; "Hierarchical Planning" and "GOMS", RARDE Project Internal Working Document Number 32.

Silcock, N., (1990a), JSD*(HF) Deliverables Corresponding to the First Conjunction with the JSD*(SE) Design Stream -- Outputs of the CTM and SUN Stages for the Security Module of the DDN NMS Case-Study, RARDE Project Internal Working Document Number 37B.

Silcock, N., (1990b), The DDN NMS Case-Study -- Documenting Outputs of the ESSA and GTM Stages for the Security Module and Associated Refinements Suggested for the JSD*(HF) Method, RARDE Project Internal Working Document Number 38B.

Silcock, N. and Lim, K. Y., (1989), Towards a Task Analysis Method for JSD* : An Analytical and Procedural Review of Task Analysis for Knowledge Descriptions and Knowledge Analysis of Tasks, RARDE Project Internal Working Document Number 31.

Silcock, N., and Lim, K. Y., (1990a), A First Pass Version of the Proceduralised JSD*(HF) Method, RARDE Project Internal Working Document Number 34.

Silcock, N., and Lim, K. Y., (1990b), The DDN NMS Case-Study -- Documenting Outputs of the CTM and Post-CTM Stages for the Security Module and Associated Refinements Suggested for the JSD*(HF) Method, RARDE Project Internal Working Document Number 39B.

Simon, T., (1988), Towards a Rational Taxonomy of Cognitive Models in Human-Computer Interaction, In : Jones, D. M. and Winder, R. (eds.), Proceedings of BCS HCI SG Conference (HCI'88 -- People and Computers IV), pp. 79-93, Manchester, September 1988, Cambridge University Press.

Smith, D. C.; Irby, C., Kimball, R., Verblank, W. and Harslam, E., (1982), Designing the STAR User Interface, Byte, April 1982.

Smith, S. L., (1986), Standards versus Guidelines for Designing User Interface Software, Behaviour and Information Technology, 1986, Vol. 5, No 1, pp. 47-61.

Smith, S. L., and Mosier, J. M., (1984), Design Guidelines for User-System Interface Software, Hanscom Airforce Base MA, USAF Electronic Systems Division, NTIS No. AD A154 907, Tech Rep ESD-TR-84-190, 1984.

Singleton, W., (1972), Introduction to Ergonomics, Geneva : World Health Organisation.

Sridhar, K. T. and Hoare, C. A. R., (1985), JSD Expressed in CSP, Technical Monograph PRG-51, Oxford University Computing Laboratory; Also in : J. R. Cameron, JSP and JSD : The Jackson Approach to Software Development, Second Edition, pp. 334-363, IEEE Computer Society Press.

Stevens, G. C., (1983), User-Friendly Computer Systems ? A Critical Examination of the Concept, Behaviour and Information Technology, Vol. 2, pp. 3-16, London : Taylor Francis.

Sutcliffe, A., (1988a), Some Experiences in Integrating Specification of Human Computer Interaction within a Structured System Development Method. In D. M. Jones and R. Winder (eds.), Proceedings of the Fourth Conference of the BCS HCI SIG Conference, Cambridge, pp. 145-160, Cambridge University Press.

Sutcliffe, A., (1988b), Jackson System Development, Prentice-Hall, London.

Sutcliffe, A., (1989), Task Analysis, Systems Analysis and Design : Symbiosis or Synthesis ?, Interacting With Computers, Vol. 1, No. 1, April 1989, pp. 6-12, Butterworth Scientific Ltd, London.

Sutcliffe, A., and Wang, I., (1991), Integrating Human-Computer Interaction with Jackson System Development, The Computer Journal, Vol. 34, No. 2, April 1991, pp. 132-142, Cambridge University Press.

Thimbleby, H., (1987), Delaying Commitment, Internal Report of the Department of Computing Science, University of York.

Thimbleby, H., (1990), User Interface Design, ACM Press New York : Addison-Wesley Publishers.

Underwood, M. J., (1987), Alvey Human Interface Club Response to the IT'86 (Bide) Report : Organisation. In : Alvey Human Interface Club, Report of Open Meeting, Strand, London, 13 October 1987, pp. 15-28.

Valusek, J. R., (1988), Adaptive Design of DSSs : A User Perspective, In : Proceedings of the 8th International Conference on Decision Support Systems, Boston, June 6-9, 1988, pp. 105-112.

van der Veer, G. C., Guest, S. and Hamilton, I., (1988), On User Modelling, In : Alvey User Modelling SIG Workshop Handout on Users' Models of Systems, 29 March 1988.

Waddington, R. and Johnson, P. (1989), A Family of Task Models for Interface Design, (Draft version), To appear in Proceedings of HCI'89 Conference.

Walsh, P., (1987a), Using JSD as a Description Language, RARDE Project Internal Working Document Number 5.

Walsh, P., (1987b), Task Analysis Methods in HCI Design, RARDE Project Internal Working Document Number 13.

Walsh, P., (1988), Proposed Plan of Action for Network Manager Case-Study, Internal Working Document Number 16.

Walsh, P., Lim, K. Y., Long, J. B. and Carver, M. K., (1989), JSD and the Design of User Interface Software. In : Barber, P. and Laws, J. (eds.), Ergonomics (Special Issue on Methodological Issues in Cognitive Ergonomics), Vol 32, No 11, November 1989, 1483-1498, Taylor & Francis London.

Whitefield, A., (1987), Models in Human Computer Interaction : A Classification with Special Reference to their Uses in Design, In : Bullinger, H. J. and Shackel, B. (eds.), Proceedings of the Second IFIP Conference on HCI - Interact'87, pp. 57-63, University of Stuttgart, Germany, 1-4 September 1987, Elsevier Science Holland.

Whitefield, A., (1990), Human-Computer Interaction Models and their Roles in the Design of Interactive Systems, In : Cognitive Ergonomics : Understanding, Learning and Designing Human-Computer Interaction, pp. 7-25, Academic Press London.

Whiteside, J., Jones, S., Levy, P. S., and Wixon, D., (1985), User Performance with Command, Menu and Iconic Interfaces. In : Borman and Curtis (eds.), Human Factors in Computing Systems II, 1985, pp. 185 - 191, North Holland.

Wigander, K., Svensson, A., Schoug, L., Rydin, A. and Dahlgren, C., (1979), Structured Analysis and Design of Information Systems, New York : McGraw-Hill Inc., 1979.

Williams, J. R., (1989), Menu Design Guidelines, ISO/WD 9241-XX (Working Draft -2), March 1989.

Wilson, J. and Rosenberg, D., (1988), Rapid Prototyping for User Interface Design. In : M. Helander (ed.), Handbook of Human Computer Interaction, pp. 859-876, Elsevier Science Publishers, North Holland, 1988.

Wilson, M., Barnard, P. J. and MacLean, A., (1986), Task Analysis in Human Computer Interaction, Report HF 122, IBM Hursley Human Factors.

Wilson, M., Duce, D. and Simpson, D., (1989), Life-Cycle in Software and Knowledge Engineering : A Comparative Review, The Knowledge Engineering Review, Vol. 4, No. 3, pp. 189-204, Cambridge University Press.

Young, R. M., (1983), Surrogates and Mappings : Two Kinds of Conceptual Models for Interactive Devices, In : In Stevens, A. L. and Gentner, D. (eds.), Mental Models, Lawrence Erlbaum Associates Publishers, Hillsdale, New Jersey.

Young, R. E., (1988), Interim Report on the COSMOS Project, Cosmos Coordinator's Office, Queen Mary College, London, Report No. 45.5 EXT.

Zave, P., (1984), The Operational versus the Conventional Approach to Software Development, Reports and Articles, Communications of the ACM, February 1984, Vol. 27, No 2, pp. 104-118, ACM Press New York.

PART VII :

Appendices

CONTENTS

Glossary of Terms.....358

Index of Abbreviations.....361

Annexes.....364

Annex A : JSD Structured Diagram Notation, JSD* Structured Diagram Notation and Task Description.....364

Annex B : Basic Design Concepts of the JSD*(HF) Method.....366

Annex C : The Hierarchical Conception of Work Assumed by the JSD*(HF) Method.....370

Annex D : Detailed Account of Secondary Activities and Products of the ESSA Stage for the Network Security Management Case-study.....374

Annex E : An Illustration of the JSD*(HF) Method using the Recreation Facility Booking System.....385

Bibliography.....413

A) List of Published Papers Originating from the Present Research.....413

B) Cumulative List of RARDE Project Working Documents.....416

C) Cumulative List of PhD Working Documents.....420

Glossary of Terms

Abstraction See Annex B.

Automated task A task performed entirely by a device.

Decomposition See Annex B.

Design phase A group of design stages that addresses a common aspect of system development. For instance, the JSD*(SE) Specification Phase (comprising JSD Modelling and Network Stages) is concerned primarily with design specification (as opposed to design implementation).

Design stage A group of design activities concerned with transforming input(s) from a preceding stage(s) into its characteristic product(s).

Device independent A term used to describe a task that is not specific to a particular device. For instance, TD(ext) is a device *dependent* description. In contrast, GTM(ext) is a predominantly device *independent* description since device-specific details were removed during its abstraction from a TD(ext) description. The purpose of deriving a device independent description is to uncover the logic underlying a design so that generalisations may be made about its functional features. Thus, the potential porting of design features across systems may be assessed. However, it should also be noted that a completely device independent description is not always desired since low level design details would be lost.

Domain objects Objects associated with the domain of application.

Extant System or EXT A general term for a class of systems comprising the extant current system, extant partial system and extant related system.

Extant System Composite or X A 'virtual' composite system synthesised analytically from a number of extant systems (either part or whole). Such a system description is derived to support subsequent reasoning about the system to be designed.

Extant Related System An existing system that shares the same domain of application as the system to be designed, but is not used by the client organisation, i.e. it is used by other organisations.

Extant Current System The existing system to be replaced by the system to be designed, i.e. the system currently used by the client organisation.

Extant Partial System Parts of an existing system that may be relevant to the system to be designed. For instance, its sub-tasks (and associated design features) may be similar to those anticipated for the system to be designed. In other words, the domains of the systems intersect partially.

Generification See Annex B.

Inter-dependencies Intersecting JSD*(HF) and JSD*(SE) design concerns. At each inter-dependency, design information of common interest would have to be shared, agreed and complied with throughout later stages of design. Since the shared information is mutually binding, unavoidable violations in one design stream must be communicated to the other. Appropriate iteration of the design stages governed by the particular inter-dependency may then be undertaken. Usually, design stages following the violation point would have to be repeated. The purpose of design inter-dependencies is to ensure that the specifications derived in parallel JSD*(HF) and JSD*(SE) streams are convergent.

Interface objects Interactive objects that may be manipulated by a user, or non-interactive components of an information display.

JSD* A term referring to the integrated method which comprises a structured human factors method (termed the JSD*(HF) method) and the JSD method (renamed the JSD*(SE) method).

JSD*(HF) The human factors component of the JSD* method.

JSD*(SE) The JSD component of the JSD* method. The JSD*(SE) method is essentially the original JSD method extended to include design inter-dependencies with the JSD*(HF) method.

Off-line task User's tasks that do not involve a computer.

On-line task User's tasks which involve direct interactions with a computer.

Screen objects A sub-set of user interface objects comprising information and

functional representations in a screen display.

Synthesis See Annex B.

System A particular configuration of human and computer entities interacting to perform work under a specific environment.

Target System The system to be designed.

Task System activity required to achieve work goals.

Index of Abbreviations

C Confirm (refers to a generic screen design template for users to confirm an input).

CTM Composite Task Model (Stage).

CTM(y) Composite task model of the target system.

DD Display Design (Stage).

DET(ext), DET(y) Dialogue and error message tables for the extant and target systems respectively.

DITaSAD(ext), DITaSAD(y) Dialogue and inter-task screen actuation descriptions of the extant and target systems respectively.

DoDD(ext), DoDD(y) Domain of design discourse description of the extant and target systems respectively.

E Error (refers to a generic screen design template used to signal error messages to the user).

ECS Extant Current System

em Error message (usually followed by a numeric suffix which identifies the message item in the Dialogue and Error Message Table).

EPS Extant Partial System

ERS Extant Related System

ESSA Extant Systems System Analysis (Stage)

EXT Extant system

(ext) Denotes JSD*(HF) descriptions of an extant system, e.g. GTM(ext).

GTM Generalised Task Model (Stage).

GTM(x), GTM(ext), GTM(y) Generalised task model of a 'virtual' extant system composite, an extant system and the target system respectively.

HF Human Factors

IM Interface Model (Stage)

IM(ext), IM(y) Interface model of the extant and target systems respectively.

ITM Interaction Task Model (Stage)

ITM(ext), ITM(y) Interaction task model of the extant and target systems respectively.

JSD Jackson System Development (method).

JSD* An integrated method comprising the Jackson System Development method and a structured human factors method.

JSD*(HF) Human Factors component of the JSD* method.

JSD*(SE) JSD component of the JSD* method.

MPASS MacPassword™ (a Macintosh-based software for PC security).

NMgr Network Manager.

NMS Network Management System.

NMW Network Management Workstation.

PSL(ext), PSL(y) Pictorial screen layout diagrams for the extant and target systems respectively.

R(ext) General term used to denote a range of products derived at the Extant Systems System Analysis Stage. Thus, STM(ext); UTM(ext); ITM(ext); IM(ext); DoDD(ext); SUN(ext); UIE(ext) and DD(ext) are all instances of R(ext).

S Screen (refers to a screen design. Usually followed by a alpha-numeric suffix which identifies the pictorial screen layout diagram for a particular screen).

SE Software Engineering

STM System Task Model (Stage)

STM(ext), STM(y) System task model of the extant and target systems respectively.

SoRe Statement of Requirements

SUN Statement of User Needs (Stage)

SUN(ext), SUN(y) Statement of user needs for the extant and target systems

respectively.

SUTaM System and User Task Model (Stage)

TD Task Description

TD(ext) Task description for an extant system.

UCL University College London

UCLCC University College London Computer Centre

UIE(ext), UIE(y) User interface environment of the extant and target systems respectively.

UTM User Task Model (Stage)

UTM(ext), UTM(y) User task model of the extant and target systems respectively.

(x) Denotes JSD*(HF) descriptions of a 'virtual' extant system composite, e.g. GTM(x).

X Represents the 'virtual' extant system composite that is derived to support design reasoning.

(y) Denotes JSD*(HF) descriptions of the target system, e.g. CTM(y).

Y Represents the target system or system to be designed.

Annexes

Annex A : JSD Structured Diagram Notation, JSD* Structured Diagram Notation and Task Description

Generally, constructs of the JSD structured diagram notation (comprising sequence, selection, iteration and posit-quit -- see Figure AA-1(A)) were recruited to the JSD*(HF) method for task description. The recruitment was motivated by two reasons, namely :

(1) design communications between JSD*(SE) and JSD*(HF) designers would be facilitated by a common notation;
(2) the JSD structured diagram notation could support a more specific description of tasks.

To this end, the utility of the notation was investigated by Cameron and Carver (1987); Walsh (1987a); Lim (1988e); and Carver (1988). Although the results were positive, an additional construct was introduced by Lim (1988e) to support the description of non-sequential hierarchies (see Figure AA-1(B) below). The application of such a construct (termed a hierarchy construct) is demonstrated in Figure AA-2 below. For comparison, JSD and JSD* structured diagram descriptions is shown for a fictitious task 'T' which may be characterised as follows :

(a) it comprises four sub-tasks, namely A, B, C and D;
(b) its sub-tasks B and C may be performed in any order, i.e. B then C, or C then B;
(c) its sub-tasks B and C can be carried out only after A;
(d) its sub-task D can be carried out only after B and C.

An inspection of the descriptions of task T reveals that the JSD structured diagram description is clearly unwieldy. In particular, the diagram would become unmanageably large with a greater number of sub-tasks. Thus, the addition of a

hierarchy construct permits a more elegant, readable and concise task description.

Figure AA-1 : Constructs of JSD and JSD* Structured Diagram Notation

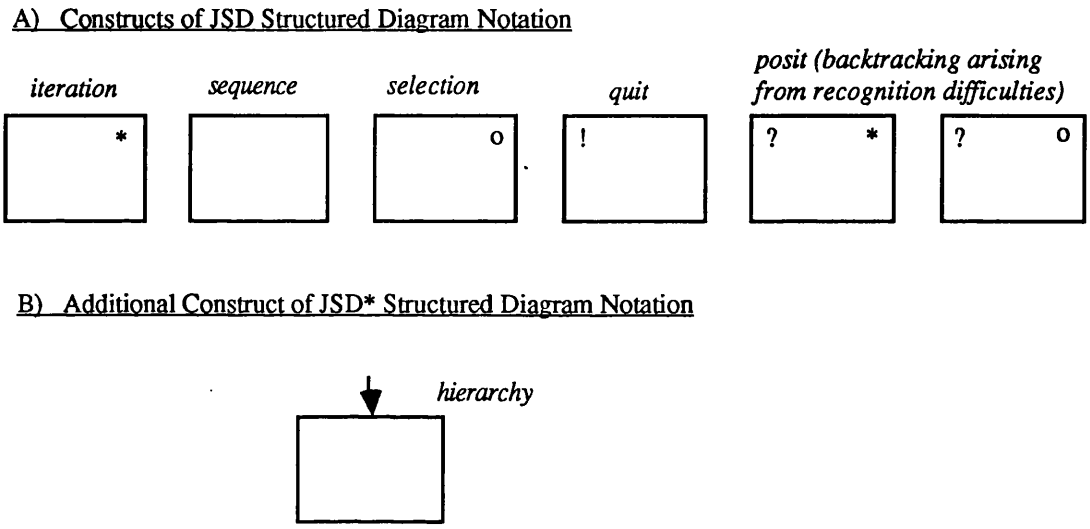
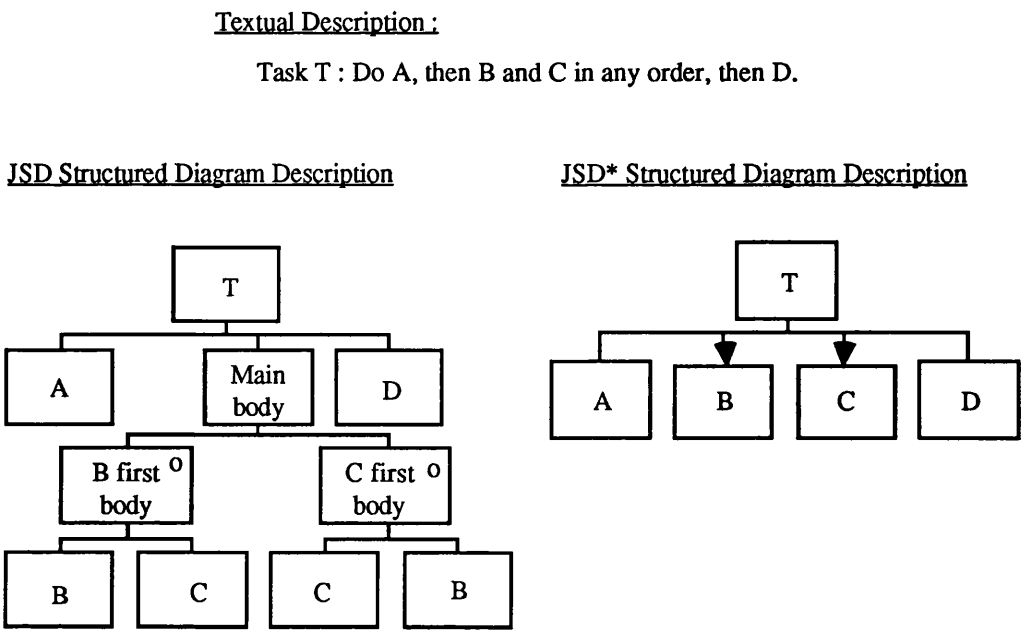


Figure AA-2 : JSD and JSD* Structured Diagram Descriptions of a Fictitious Task 'T'



Annex B: Basic Design Concepts of the JSD*(HF) Method

(a) Abstraction

Abstraction is applied by the designer to derive a particular perspective of a task. For instance, task performance requirements may be viewed either in terms of user operations with a device, or in terms of higher level sub-task units performed to satisfy specific system goals.

In the context of the JSD*(HF) method, abstraction is used primarily to expose the conceptual (rather than detailed) design of a particular task. Specifically, it supports the derivation of generalised task models of the extant and target systems, i.e. GTM(ext) and GTM(y) respectively.

(b) Generification

Generification is a process that identifies common elements among discrete objects so that they may be characterised as a set for a specific design context. In particular, the common elements define a super-ordinate or 'generic' class to which all the original objects would belong. As an example, consider objects A, B and C whose attributes are represented by the sets {1, 2, 3, 4}, {1, 2, 3, 6, 7} and {1, 2, 3, 8, 9} respectively. Thus, for a specific design context, a generic class G comprising the intersection of their attributes, i.e. G = {1, 2, 3}, may be defined to characterise objects A, B and C as set. It can not be over-emphasised that the selected set of generic attributes is determined largely by the design context.

In respect of the JSD*(HF) method, generification is used in the following :

(i) the identification and selection of extant systems for analysis. In this case, the set of generic attributes of interest corresponds to the key characteristics of the target system. By selecting particular sub-sets of these characteristics, one or more generic categories may be defined to guide the selection of extant systems for analysis. For an illustration, consider the

generic class G from the above example. In this case, G describes a generic set of target system characteristics of interest, and A to C characterise extant systems selected for further analysis;
(ii) the derivation of a single Task Description (TD(ext)) from information elicited from various sources. In other words, a generic task description is constructed using task elements that are common across performers. Thus, a super-ordinate description applicable across task performers is derived. Taking the above example further, A, B and C may comprise various accounts of a task as described by different performers. A generic task description, considered by the group of performers to be equivalent to their original account, may then be derived to support system design. Such a description may be characterised by H = {1, 2, ??), where ?? was considered by different task performers to be equivalent to {4}, {6, 7} and {8, 9}.

Four other scenarios for applying generification are described in Figure AB-1 below. A brief account of these scenarios follows.

First, generification *across sub-task constituents* may be applied on *a* task description elicited from *a* task performer. In this case, generification may be applied to remove inconsistent sub-task level descriptions. To this end, a

Figure AB-1 : Scenarios for Applying Generification[1]

Scenario	Generification Type
Single task performer and single task	Across objects, actions and attributes
Many task performers and single task	Across task performers
Single task performer and many related tasks	Across tasks
Many task performers and many related tasks	Across task performers and/or tasks

[1] Note that generification across sub-tasks is possible in all scenarios.

generic set of attributes, actions and objects is defined, and the sub-task is then re-described in terms of the set.

Second, generification *across tasks* may be applied when *a* task performer describes *many related* tasks that share a common underlying 'logic'. Depending on the design context, it may be useful to derive a generic description of aspects that are common across the tasks.

Third, generification *across task performers* may be applied when *a* task is described by *many* task performers. In such cases, it is necessary to derive a single task description with which to work during design. For instance, the descriptions of task objects and their manipulations may vary across task performers. If it is established that the variations are not due to different task performance strategies, they should be removed by generification.

Fourth, generification *across task performers and/or tasks* may be applied when *various* related tasks are described by *different groups* of task performers. An example of such an application was reported by Johnson, Diaper and Long (1984) concerning training syllabus development for groups of students who are required eventually to perform slightly different tasks. Their motivation for applying generification is as follows :

> (1) to define a common training component that satisfies task requirements that are generic across the student groups;
> (2) to define separate and distinct training components that satisfy task requirements unique to each student group.

To this end, a generic description was derived to characterise each group of students and their tasks. The generic descriptions were then compared to identify common and specific knowledge requirements for performing the tasks. Appropriate training syllabi were thus developed for each group of students.

In conclusion, it should be noted that the selection of a particular application of

generification is determined by the prevailing design context.

(c) Decomposition

Task decomposition is characterised by a successive breakdown of tasks to the level of description required to support design. In particular, task decomposition is used in the JSD*(HF) method for the following :

(1) to derive a task description for the extant system, i.e. TD(ext);
(2) to generate detailed specifications from a conceptual design.

(d) Synthesis

Task synthesis addresses the appropriate composition and extension of sub-tasks so that system goals are satisfied at acceptable cost. In the context of the JSD*(HF) method, task synthesis includes the selection, extrapolation and incorporation of suitable sub-tasks of extant systems. Consequently, task synthesis is undertaken only after abstraction and generification. In particular, the latter processes are applied extensively during extant systems analysis to uncover the logic of extant designs. Thus, appropriate extant system features may be identified for recruitment to the target system.

Task synthesis is undertaken predominantly at two stages of the JSD*(HF) method, namely :

(a) the Generalised Task Model Stage where suitable sub-sets of individual GTM(ext) descriptions are composed to generate a GTM(x) description;
(b) the Composite Task Model Stage where a suitable sub-set of the GTM(x) description is composed with a GTM(y) description. The integrated description is then extended to generate a CTM(y) description.

Annex C : The Hierarchical Conception of Work Assumed by the JSD*(HF) Method

Figure AC-1 shows the hierarchical conception of work proposed by Lim (1989d) following a review of similar attempts by Johnson et al (1988), Waddington and Johnson (1989), Blyth and Hakiel (1989), and Dowell and Long (1989). Thus, the JSD*(HF) method (and human-computer interaction in general) may be set against a broader conception of the work context. In this way, user tasks, roles, jobs, etc. may be considered appropriately during design. An account of the conception follows :

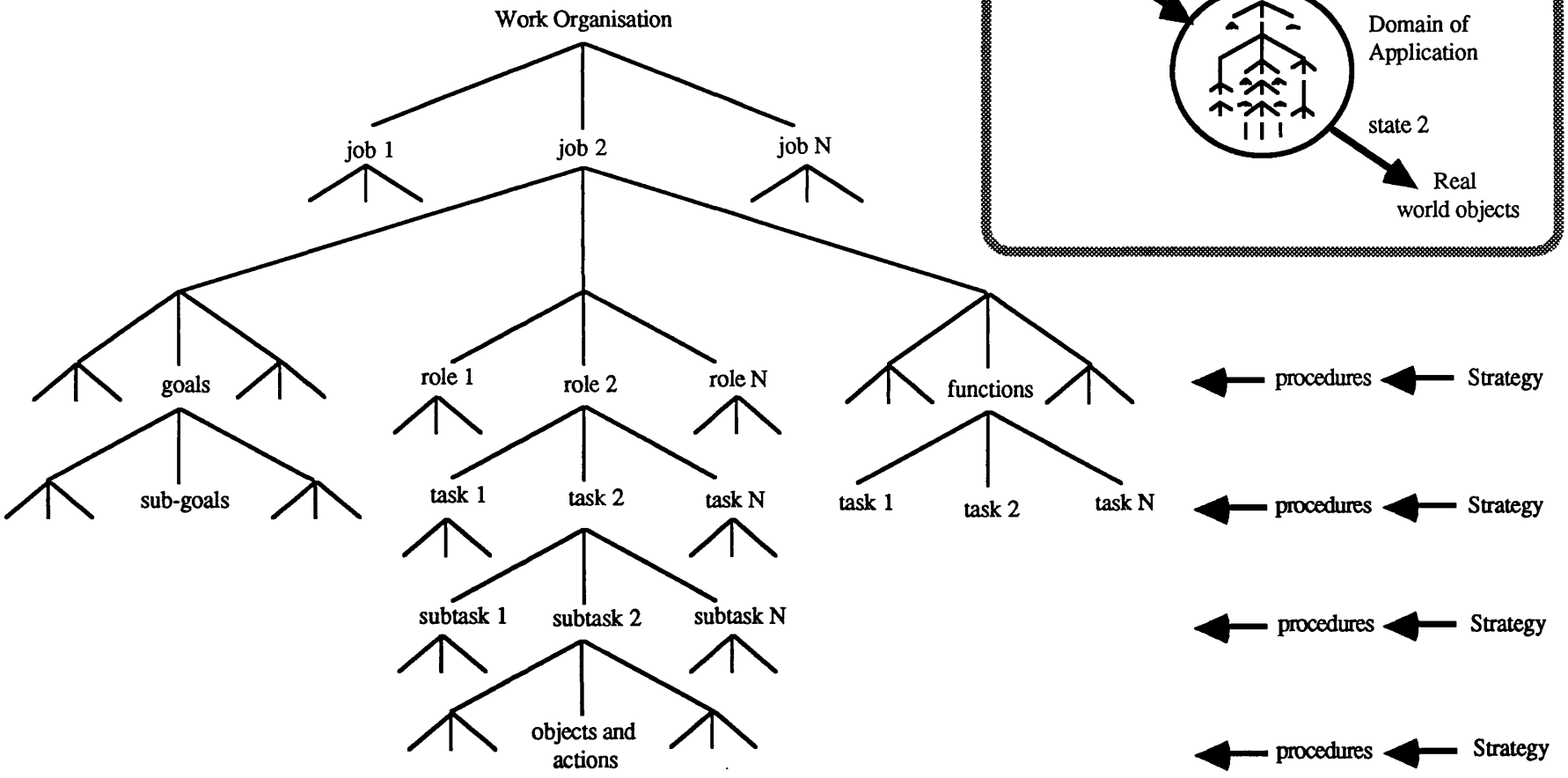
(a) the highest level in the conception is the organisation tasked with performing the work. As such, an organisation may be conceptualised as a 'superordinate' system that may be decomposed into a number of sub-systems. In other words, a job involving the performance of one or more roles (or groups of tasks) is assigned to each sub-system. The tasks are performed by the sub-system in accordance with a plan. Such a plan comprises task execution procedures and strategies that determine the application of specific procedures;

(b) the conception is decomposed into three branches that support complementary perspectives on a design. For instance, the object and action branch supports specific design description, while design descriptions based on the goal and function branches may reveal the logic underlying particular characteristics of task performance (see Figure AC-1);

(c) procedures and strategies of the conception may apply at several levels of description, namely at the objects and actions level; goals and sub-goals level; tasks and sub-tasks level; and functions level. Thus, task procedures and strategies may interact across levels of description;

(iii) roles are viewed in the conception as comprising a particular group of functions (see Figure AC-1). In this respect, Human Factors and Software Engineering perspectives of functions may differ slightly. In Human Factors, a function may be considered a non-trivial unit of behaviour (human or device) that enables the accomplishment of the system 'mission' (Drury, 1983). Thus, a function may be allocated, *as a whole*, to a human

Figure AC-1 : The Hierarchical Conception of Work Assumed by the JSD*(HF) Method



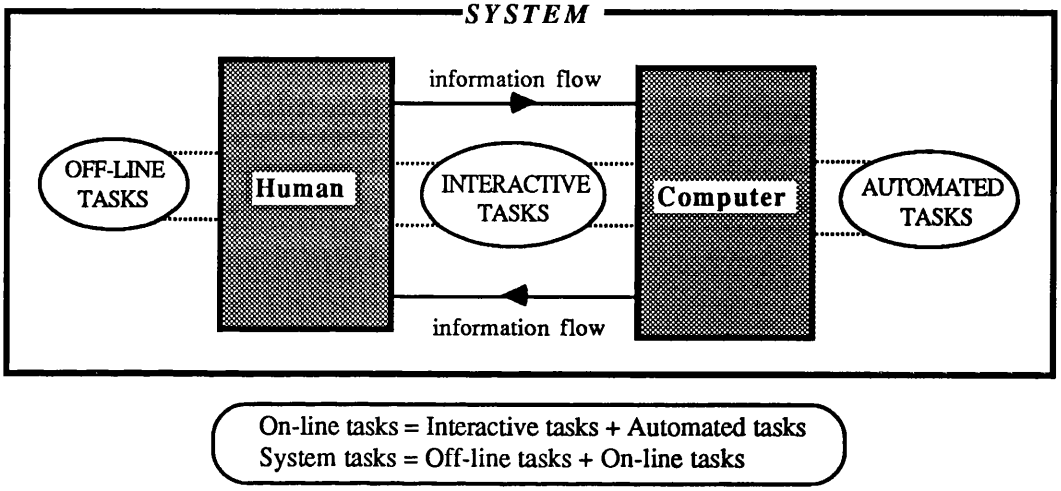
or device. In contrast, Software Engineering functions tend to refer to low level functions supported by the computer software. Nevertheless, the differences would be reconciled if functions are considered to include tasks that do not necessarily result in work (see (iv) below);

(iv) the conception defines work as the achievement of desired state changes for a relevant set of real world objects. Similarly, work goals and sub-goals may be described in terms of the initial-to-target state changes to be effected by the system. In this respect, a system (as opposed to a computer system) is defined as comprising a human-and-computer 'unit' working together under a specific environment.[2] A distinction should also be made between the work the system is expected to accomplish, and the tasks performed by the human and computer components of the system. In particular, since work goals and tasks do not necessarily share a one-to-one relationship (see Figure AC-1), task executions may not always *produce* work, i.e. they include actions which *facilitate* work.

(v) the conception classifies tasks into off-line and on-line tasks. Generally, on-line tasks involve interactions between the human and computer, while off-line tasks are unsupported by the computer. In this context, on-line task components may be assigned entirely to the computer, i.e. automated tasks. Such a classification of tasks is consistent with proposals made by other researchers. For instance, Johnson and Johnson's (1988) classification comprises tasks which are *fully, partially* and *unsupported* by the computer. Similarly, Sutcliffe (1988a) suggested three categories of tasks, namely *non-computerised* tasks; *fully computer-supported* tasks; and *interactive* tasks. Figure AC-2 shows a 'consensus' classification of tasks and their relationship to the components of a system.

[2] Note that the definition includes multiple human-and-computer 'units'.

Figure AC-2 : Task Classes and their Relationship to Human and Computer Components of a System



Annex D : Detailed Account of Secondary Activities and Products of the ESSA Stage for the Network Security Management Case-study

This annex completes the account of the ESSA Stage described in Chapter Nine. In particular, secondary activities and products relevant to a variant design scenario are described. Note that the extent to which such activities and products are accommodated, depends largely on how similar the characteristics and implementation technology are between the extant and target systems. Bearing this point in mind, the complete range of secondary products and activities of the ESSA Stage will now be described. Illustrations from the Network Security Management case-study are included where appropriate.

(I) Extant System Task Model (STM(ext)) and Extant User Task Model (UTM(ext)) Descriptions

To characterise conceptually the on-line task of the extant system, a description termed an extant System Task Model (STM(ext)) is derived. Thus, the on-line task is described in terms of human-computer interaction cycles. Since STM(ext) is a high level description, the form of the interaction should not be specified. Such design details are accommodated later by other ESSA Stage products (refer to later accounts on ITM(ext), IM(ext) and DD(ext)).

A complementary description termed a User Task Model (UTM(ext)) is also derived to document the off-line task of the extant system. Thus, off-line sub-tasks that may be relevant to the design of the target system are collated into a single description.

The procedures for deriving STM(ext) and UTM(ext) descriptions are summarised overleaf.

Procedures for deriving STM(ext) and UTM(ext)

1. Take GTM(ext) as input and identify on-line and off-line tasks. If more detailed information is desired, derive a lower GTM(ext) description by consulting TD(ext). If necessary, conduct another elicitation cycle.

2(a) To derive STM(ext), note user actions and computer responses for each on-line sub-node of the GTM(ext) description. Thus, the H: (human) and C: (computer) leaves of the STM(ext) description are derived.

2(b) UTM(ext) is derived by noting off-line sub-nodes of the GTM(ext) description. The sub-nodes are then collated to derive a single description.

3. Record STM(ext) and UTM(ext) descriptions using JSD structured diagram notation and note additional information in a supporting table.*

Rules of Thumb for deriving STM(ext) and UTM(ext)

1. Where it is uninformative to describe on-line sub-tasks in terms of separate H: and C: leaves, then combined (i.e. H-C:) nodes may be used.

2. The structure of the GTM(ext) description should be maintained in STM(y) and UTM(y) (if possible) to facilitate cross-referencing between the descriptions. In addition, a more coherent view of the task is afforded.

3. For the same reason as in (2) above, significant off-line should also be indicated in the STM(ext) description. Thus, cross-referencing between STM(ext) and UTM(ext) descriptions is supported. Similarly, off-line tasks which influence the design of the user interface should be flagged in the STM(y) description to prompt appropriate consideration by the designer.

(II) Extant Interaction Task Model or ITM(ext) Description

ITM(ext) is a device-level description of the interactive task to be performed by the user. It is derived by decomposing H: leaves of the STM(ext) description.

The procedures for deriving an ITM(ext) description are summarised on the next page.

Procedures for deriving ITM(ext)

1. Take each H: leaf of the STM(ext) description and note how inputs are effected via the user interface. Consult the TD(ext) description for more information if necessary. A further elicitation cycle should be conducted if required.

2. On the basis of the notes made in (1) above, decompose H: components of the STM(ext) description to derive an ITM(ext) description. The description details the user inputs required to perform the interactive task.

3. Record the ITM(ext) description using JSD structured diagram notation and note user problems and additional information in a supporting table.*

Rule of thumb for deriving ITM(ext)

ITM(ext) components attributed to the adopted user-interface environment need not be described since they should be familiar to design team members.

(III) Extant Interface Model or IM(ext) Descriptions

IM(ext) descriptions address the appearance and behaviour of user interface objects of the extant system. The descriptions are decomposed from the C: leaves of the STM(ext) description.

Procedures for deriving IM(ext) descriptions are summarised below.

Procedures for deriving IM(ext)

1. Identify user interface objects of the extant system systematically by referring to the C: leaves of the STM(ext) description.

2. Using JSD structured diagram notation, record the appearance and behaviour of each object. In addition, note object responses to user inputs described in ITM(ext).*

Rule of thumb for deriving IM(ext)

User interface objects originating from the adopted user interface environment need not be described since they should be familiar to design team members.

(IV) Extant Display Design or DD(ext) Descriptions

DD(ext) descriptions comprise the following set of products :

(a) an extant Dialogue and Inter-Task Screen Actuation Description or DITaSAD(ext). The context for actuating screen displays and for triggering dialogue and error messages is specified by this description;

(b) extant Pictorial Screen Layout or PSL(ext) descriptions. Static aspects of screen displays, e.g. composition, layout and grouping of screen objects are specified by these descriptions;

(c) an extant Dialogue and Error Message Table or DET(y). Screen messages described in DITaSAD(ext) are indexed in this table.

Procedures for deriving DD(ext) descriptions are summarised below.

Procedures for deriving DD(ext)

1. *Take as input ITM(ext), IM(ext) and TD(ext) descriptions. Note the screen designs and transitions in relation to major executions of the user's task. Relate the transitions to actions described in ITM(ext), and note the ITM(ext) leaves bounded by each screen transition. Thus, screen transitions are mapped onto specific user tasks.*
2. *Assign a unique number to each screen to support cross-referencing among ITM(ext), IM(ext) and DD(ext) descriptions.*
3. *For each screen, note potential user errors and the error messages displayed. In addition, note information displays. Assign a unique identifier to each message item and tabulate them as per the DET(ext) format.*
4. *Referring to notes made in (1) and (3) above, construct a DITaSAD(ext) diagram. Conditions that should be satisfied for each screen transitions should be included in the structured diagram and its supporting table.*
5. *On the basis of notes made in (1) and (3) above, document existing screen designs which may be relevant to the design of the target system. User problems and the rationale underlying existing designs should also be documented.*

(V) Extant Statement of User Needs or SUN(ext) Description

Pertinent human factors observations made during extant system analysis are summarised textually in a SUN(ext) description. Generally, extant system information of interest would include the following :

(a) user problems with the extant design;
(b) extant design rationale, requirements and constraints;
(c) possible solutions to observed user problems;
(d) user needs not supported by the extant design.

Relevant SUN(ext) information is carried forward to the Statement of User Needs Stage where they are synthesised with the initial statement of requirements for the target system. The set of requirements is then extended appropriately to define a basis for target system design.

Procedures for deriving a SUN(ext) description are summarised below.

Procedures for deriving SUN(ext)

1. Referring to key target system requirements, construct a list of general design concerns and user needs to be investigated during extant systems analysis. The list may be incremented subsequently to account for early observations and products of extant systems analysis.

2. For each item on the list, note design information that is potentially relevant to the design of the target system, e.g. existing design rationale and constraints; user problems; etc.

3. Record suggestions on how existing problems may be obviated. The suggestions should be interpreted in the context of the target system.

4. Collate the information following the format of a SUN(ext) description.

Case-Study Illustrations of a SUN(ext) Description

Case-study illustrations of SUN(ext) descriptions for the University of London

Computer Centre (UCLCC) and the MacPassword™ application (MPASS) are shown in Figures AD-1 and AD-2 respectively.

An account of how SUN(ext) descriptions could support target system design follows. The information in SUN(UCLCC) suggested that failed log-on events should be signalled to the network manager in real time when the network management workstation is manned. Should such events occur when the workstation is unmanned, the manager should be alerted on next log-on. In either case, security alerts should include pertinent information on the event to support the manager's decision concerning an appropriate response. An example of the information to be included was described in SUN(MPASS), namely a password-and-time log to differentiate between password mis-types and hacking attempts. These SUN(UCLCC) and SUN(MPASS) recommendations are thus considered during target system design.

Figure AD-1 : Extract from the Extant Statement of User Needs for the University College London Computer Centre (SUN(UCLCC))

Statement of User Needs (SUN(UCLCC))

1. The network manager should be alerted in real-time to failed log-on events. The alert should adequately capture the network manager's attention (consider both visual and auditory alarms). Should the breach occur when the network management workstation is unmanned, the manager should be alerted on next log-on.

2. Communication between the network manager and users (and between the manager and external networks) should be supported adequately. In particular, facilities that support both synchronous and asynchronous communication should be provided, e.g. telephone and electronic-mail facilities respectively.

3. Information processing functions should accompany security alerts. Thus, relevant information may be collated to support various task contexts. In particular, the functions should support database search and retrieval so that various logs may be collated, e.g. logs of atypical network usage (e.g. usage outside normal working hours); log of previous hacking activities (i.e. both failed and illegal log-ons); etc.

4. The number of successive failed log-ons should be limited to ensure a more secure network. On reaching such a limit, a temporary 'lock out' should be enforced.

.....etc.

Figure AD-2 : Extract from the Extant Statement of User Needs for the MacPassword™ Application (SUN(MPASS))

Statement of User Needs (SUN(MPASS))

1. The owner is required to search event logs for evidence of security breaches. To support the task, appropriate search functions should be provided to highlight pertinent information in the logs.

2. Password logs were found to be useful in identifying hacking activity. Thus, passwords used in failed log-ons should be recorded.

3. Obligatory password changes should be enforced at regular intervals.

4. User access parameters should be configurable and implemented easily.

5. Secondary measures to protect event logs from hacker access should be considered since no other record of usage activity is kept.

.....etc.

(VI) Extant Domain of Design Discourse or DoDD(ext) Description

DoDD(ext) is a semantic description of the extant system domain. The domain of extant and target systems intersect to different extents depending on the type of extant system involved. Specifically, the extent of domain intersection with the target system decreases in the following order : extant current system, extant related system and extant partial system.

Procedures for deriving a DoDD(ext) description are given on the following page.

Case-Study Illustrations of a DoDD(ext) Description

As before, two case-study examples of a DoDD(ext) description are shown, namely DoDD(UCLCC) and DoDD(MPASS) (refer to Figures AD-3 and AD-4 respectively). Design information tables that accompany these descriptions are also shown.

Procedures for deriving DoDD(ext)

1. Identify general design concepts and real world objects manipulated by the extant system.

2. From the set in (1) above, identify a sub-set that is potentially relevant to the target system. Define the semantics and relationships among the design concepts and objects.

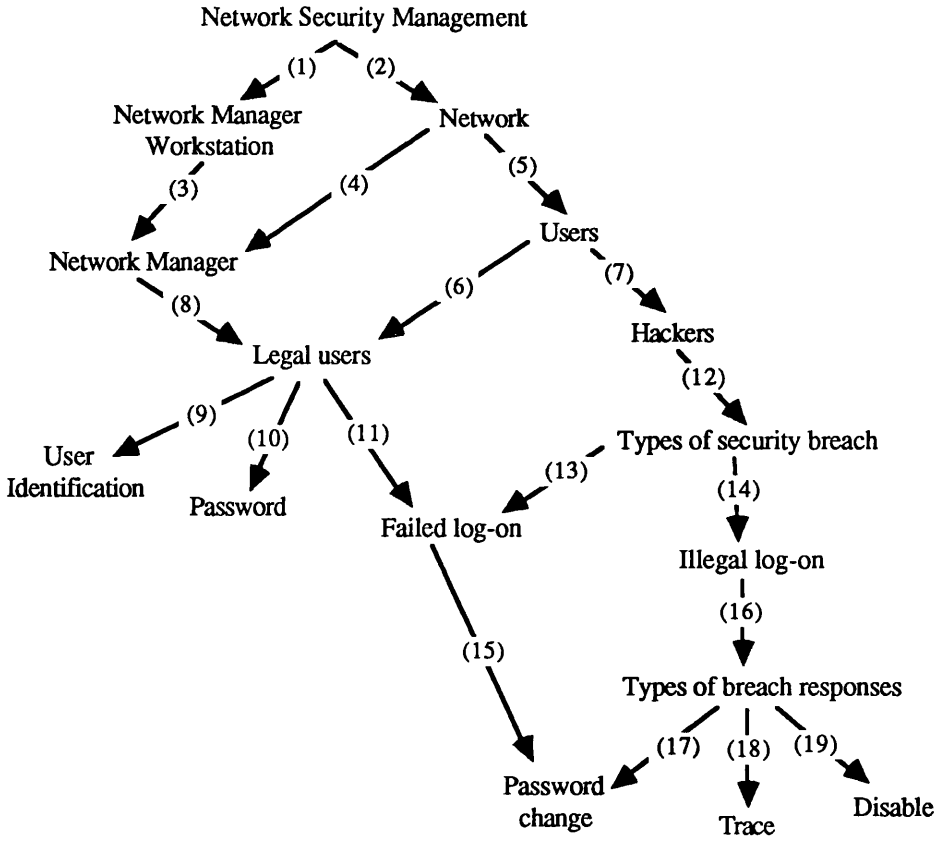
3. Record the information as a semantic net, i.e. the format of a DoDD(ext) description. Assign unique identifiers to all relations. For readability, the identifiers should be assigned serially from left to right and top to bottom (where possible). Note additional information on each relation in an accompanying table.

Rules of thumb for deriving DoDD(ext)

1. DoDD(ext) should be sufficiently detailed to support the construction of task scenarios that are accommodated by the extant system.

2. DoDD(ext) should not include device dependent information. The information described should be confined to the semantics of the extant system domain.

Figure AD-3 : Domain of Design Discourse Description for the University College London Computer Centre (DoDD(UCLCC))



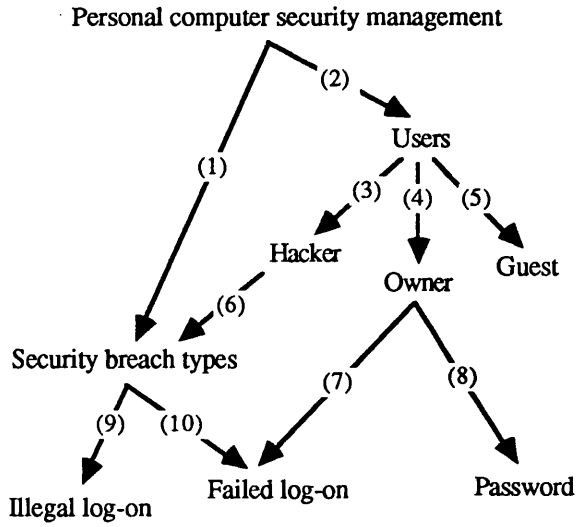
DoDD(UCLCC) Table -- Page 1

Node	Description	No.	Relations
Network Security Management	The network manager is responsible for ensuring legal access and usage of the network and network management workstation.	(1)	The task of ensuring the security of the network management workstation is minor, since it can not be accessed remotely by network users. In any case, remote access has to be cleared specifically by the computer centre.
		(2)	The network manager's main task is ensuring the security of the network.
NMW	The Network Management Workstation (NMW) is the computer system used by the manager to control the network.	(3)	The network manager is the main user of the network management workstation.
Network	The network comprises a number of host machines and terminals.	(4)	The network manager may remotely access and control the network.
		(5)	The network is open to a range of users who may log-on locally or remotely via other terminals, networks and dialling lines.
Users	Users access the network to use its services.	(6)	Only legal users are authorised to access the network and/or network management workstation.
		(7)	A hacker is a user who does not have authority to access the network and/or network management workstation.
Legal users	Users who are authorised to use the network and/or network management workstation.	(8)	The network manager is a legal user of the network and network management workstation.
		(9)	A legal user is issued with a user identification.
		(10)	A legal user is issued with a password (a confidential identification).
		(11)	Legal users may commit failed log-ons (e.g.password mis-type).
Hackers	Hackers are unauthorised users of the network and/or network management workstation.	(12)	Hackers commit security breaches.
Types of security breach	There are two types of security breach.	(13)	Failed log-on.
		(14)	Illegal log-on.

DoDD(UCLCC) Table -- Page 2

Node	Description	No.	Relations
Failed log-on	Occurs on incorrect user identification or password input. The occurrence terminates the session in all cases.	(15)	In response to a failed log-on by a legal user, the network manager may enforce a password change to encourage the selection of a more suitable password.
Illegal log-on	An illegal log-on occurs when a hacker successfully logs onto the network and/or network management workstation.	(16)	Appropriate actions must be taken in response to an illegal log-on.
Types of breach responses	Three actions may be taken in response to security breaches.	(17)	The user's password may be changed.
		(18)	Attempts may be made to trace the hacker.
		(19)	The user identification may be disabled if the user cannot be contacted to verify the status of a security breach event.

Figure AD-4 : Domain of Design Discourse Description for the MacPassword™ Application (DoDD(MPASS))



DoDD(MPASS) Table

Node	Description	No.	Relations
Personal computer security management	The personal computer owner is responsible for managing its security.	(1)	Security management involves searching logged usage information.
		(2)	The objective is to ensure that personal files are accessed only by the owner.
Users	There are three categories of users.	(3)	A hacker is an unauthorised user who attempts to break into the computer system.
		(4)	The owner is an authorised user, responsible for ensuring computer system security.
		(5)	A guest is an authorised user who may access files selected at the owner's discretion.
Hacker	A hacker is an illegal user who is intent on breaking into the computer system.	(6)	A hacker may be responsible for two types of security breach : illegal log-on (9); and failed log-on (10).
Owner	The owner is responsible for securing the computer system and for taking actions on a security breach.	(7)	A failed log-on can be attributed to a password mis-type by the owner.
		(8)	The owner has a password to access the computer system.
Types of security breach	There are two types of security breach.	(9)	An illegal log-on is a successful attempt by a hacker at breaking into the computer system.
		(10)	A failed log-on is an unsuccessful attempt by either a hacker or the owner at accessing the computer system.

On the basis of key target system characteristics, a relevant sub-set of the above DoDD(ext) descriptions is selected and synthesised. The sub-set is then extended at the Statement of User Needs Stage to generate a Domain of Design Discourse Description for the target system.

Annex E : An Illustration of the JSD*(HF) Method using the Recreation Facility Booking System (Adapted from Lim et al, 1992)[3]

The Recreation Facility Booking System supports two main activities, namely:

(a) it allows authorised users to make advance bookings of recreation facilities on a first-come-first-served basis;
(b) it permits the checking of all bookings made during the current day, and up to a week in advance.

The design scenario of the case-study is a re-design of the Recreation Facility Booking System, i.e. a variant design scenario. The motivation for the re-design is to provide more effective support to users.

The simplicity of the Recreation Facility Booking System is exploited presently to illustrate the stage-wise products of the method.

(I) The Extant Systems System Analysis Stage

The objective of this stage is two fold:

(a) to generate background information for design, e.g. capturing details of the current system. The information of interest includes current user needs and problems, the current task, existing design features and rationale, etc.;
(b) to assess extant design characteristics that may potentially be recruited to the target system.

In addition to supporting target system design, the acquired extant systems

[3] The paper was written in mid-1990. A revised version was submitted for publication around March 1991. Although the present version is a revision of the 1991 version, minor differences with the latest version of the method (described in Chapters Nine to Eleven) were retained for comparison.

information may subsequently support assessments of possible transfer of learning by end users from the extant system to the target system (both positive and negative transfer effects). An account of the high level processes for achieving these objectives follows.

To initiate extant system analysis, appropriate extant systems are identified on the basis of key target system characteristics extracted from the initial statement of requirements (collated from the client's brief and contractual documents). Extant design information is then elicited from various end-user groups using 'off-the-shelf' elicitation techniques, e.g. structured interviews, unobtrusive observations, etc. As a guide, an acceptable level of task description should satisfy two rules of decomposition, namely :

(a) the resulting description should be understood by designers and end-users;
(b) task decomposition should be terminated only when an acceptable Probability of failure x Cost (or P x C) criterion is reached (Duncan, 1974).

In most cases, the information would be elicited from various sources, e.g. from different end-users and literature sources. Thus, a set of generic descriptors should be derived to organise sensibly the information from these sources, i.e. 'building blocks' that are common across the sources have to be identified. For this purpose, the generification procedures suggested by Johnson and Johnson (1987); and Johnson, Johnson and Russell (1988), have been adapted for the JSD*(HF) method.[4] In addition, device dependent characteristics have to be removed by abstraction to facilitate subsequent comparisons between extant and target system designs. To this end, extant system descriptions should be abstracted to a sufficiently high level to reveal the logic underlying their designs. However, the level of abstraction need not be homogenous throughout the structured diagram description. In particular, to preserve information on extant design features that are potentially relevant to the design of the target system, the level of abstraction may be deliberately lower for specific parts of the description.

[4] See also Footnote 5.

Since the information derived from extant systems analysis contributes to subsequent stages of the JSD*(HF) method, it is processed into products that correspond in scope and format to products of these stages (see later). Thus, an appropriate account of extant systems information is facilitated. Depending on the prevailing design circumstances (e.g. in the case of variant design similarity between extant and target systems is expected to be high), products derived at the Extant Systems System Analysis Stage may range from the extant task description, and generalised task model to the full JSD*(HF) complement. In other words, a full complement of descriptions would include the system task model, user task model, interface model, interaction task model and display design descriptions (see Figure 8-1). (Note that the extant task description is a dynamic representation of task execution. Other products will be characterised later). In addition to the prevailing design circumstances, the extent to which extant systems analysis is undertaken depends largely on the designer's familiarity with the system domain, i.e. how well defined the target system is.

Superficially, the outputs of this stage may seem to support only variant design (which incidentally accounts for the vast majority of system development projects, see Rouse and Boff, 1987) as opposed to 'original' or 'novel' design. Thus, it should be emphasised that the method is not limited to variant design. Together with the procedures of the Generalised Task Model and Composite Task Model Stages, 'original' or 'novel' design is also supported (see later).

Products of extant systems analysis are documented using a combination of text, JSD* structured diagram notation and design information tables. Figure AE-1 shows an extant task description for the case-study, described using JSD* structured diagram notation. Further information on the nodes and leaves of the description are expanded in an accompanying table as shown in Table AE-1.

Figure AE-1 : Extant Task Description for the Recreation Facility Booking System

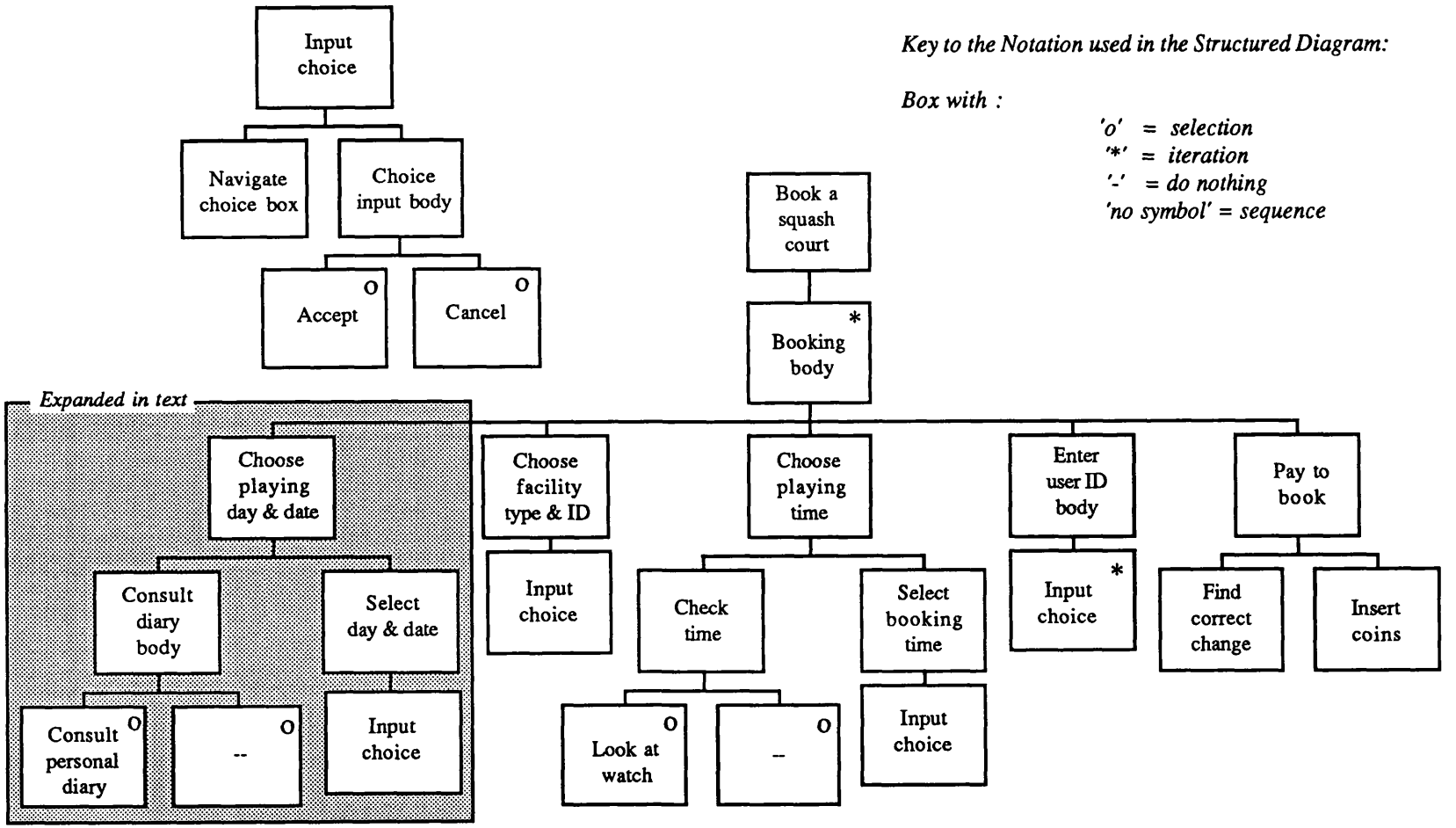


Table AE-1 : Extant Task Description Table for the Recreation Facility Booking System

Node	Description and Observations	Design Implications and Speculations
Book a squash court	The user can book recreation facilities and check previous reservations.	
Booking body	The user inputs booking parameters one at a time in the following sequence : day & date, facility type and playing time. Long search time requiring many search cycles.	Poor design of search tasks. Should adopt conjunction rather than sequential search for day & date, and playing time parameters.
Choose playing day & date	The user may consult a diary before deciding on a recreation day & date. High proportion of 'quit' actions at this stage. The design does not allow the user to confirm inputs before computer implementation.	Provide a separate 'confirm' step to permit user checking.
....etc.		

The Figure and Table will now be described in greater detail. Figure AE-1 shows the interactive task imposed on users of the current Recreation Facility Booking System. Essentially, users are required to select booking parameters sequentially (i.e. day and date, time slot, recreation type) by navigating a cursor over available parameter values. These values are matched iteratively by the user against a diary of appointments. Problems with the current design are documented as shown in Table AE-1. In particular, overly long trial-and-error matching was observed and attributed to the limited amount of information that could be presented on an eight-line LCD screen. Lessons learnt from extant systems analyses are then carried forward to the next stage of JSD*(HF) design.

(II) The Generalised Task Model Stage

The objectives of this stage comprise the following :

(a) to generate predominantly device independent descriptions[5] that support comparative mapping and evaluation between extant design features and target system requirements. Thus, abstracted extant task descriptions derived in the preceding design stage are synthesised into a 'Generalised Task Model of the extant system or *X*' (termed a GTM(x) description).[6] On the basis of a GTM(x) description, appropriate extant designs may be ported to the target system;

(b) to expose new and/or salient characteristics destined for the target system by extracting a 'Generalised Task Model of the target system or *Y*' (termed a GTM(y) description) from the statement of requirements. Thus, the scope and conceptual structure of the target system is characterised. On the basis, extant systems (part or whole) may be identified for analysis. For instance, GTM(y) may indicate that authorisation to use the recreation facilities is to be controlled by booking system access. Access control design is thus included in the extant systems survey, e.g. control by identity cards (current system design); passwords (related system design); and magnetic strip-cum-personal identification number (related system design).

The two generalised task models are carried forward to the Composite Task Model Stage where desirable and compatible elements of GTM(x) are synthesised with GTM(y) on the basis of the statement of user needs (see later). The result is a composite task model of the target system. Note that the generalised task models

---

[5] The extent of abstraction to device independence is determined by how dissimilar extant system characteristics are from key target system characteristics. The selected level must be sufficiently high to reveal semantic and logical requirements of the task (as opposed to device dictated requirements, e.g. particular keystrokes), and still retain specific extant system information of interest to the design of the target system.

[6] Synthesis of abstracted task descriptions is only necessary if more than one extant system had been analysed, e.g. if both related and current systems had been analysed. If the current system is the only system analysed, then GTM(x) is the same as the GTM(current system).

could provide an early indication of the extent of training required by target system users (corresponding to the complexity of GTM(y)), and the transfer of learning that may arise (corresponding to the extent of porting from GTM(x)).

The products of the Generalised Task Model Stage are documented using JSD* structured diagram notation. A case-study example will not be given since a generalised task model is essentially similar in nature to a composite task model (an example of which is shown in Figure AE-3).

(III) The Statement of User Needs Stage

The purpose of the Statement of User Needs Stage is to augment the initial statement of requirements in respect of user needs for the target system. Thus, conclusions drawn from analysing extant designs are summarised with respect to the user. Enhancements of the initial statement of requirements may comprise the following :

(a) identification of user problems with the extant system;
(b) a summary of the rationale, requirements and constraints underlying the extant design. Such a summary supports a deeper analysis of user problems and needs. In addition, it supports the identification of promising extant designs for porting to the target system. The latter objective is thus consistent with Newman's (1988) suggestion on maximising the re-use of established user interface styles;
(c) a more explicit expression of design criteria (such as performance requirements) for obviating the problems in (a), and for upgrading or extending existing design solutions identified in (b).

The resulting JSD*(HF) description constitutes a human factors view of target system requirements.

As an example, end-user problems and potential design solutions such as those documented in Table AE-1 may be analysed and assessed against target system

requirements. Thus, it may be concluded that better search functions should be provided on the target Recreation Facility Booking System. Alternatively, a larger display screen may be specified as a prerequisite for faster and less onerous selection of a suitable booking slot (a larger screen would support conjunctive search since booking information on more than one booking parameter could be displayed at each user-computer transaction). These user requirements are then collated textually to generate a SUN(y) description (not shown for the case-study).

In addition to SUN(y), a second product is derived at the Statement of User Needs Stage. The product, termed a Domain of Design Discourse or DoDD(y) description, establishes the domain semantics required for interpreting design and task concepts associated with the target system. For instance, the domain of the Recreation Facility Booking System may be characterised as the assignment of temporal rights over a recreation facility. Figure AE-2 shows how the relationship between DoDD(y) entities may be described explicitly using a semantic net.[7] In this way, a textual account of the nodes and relations of the semantic net is detailed in a separate table (see Figure AE-2).

These two products of the Statement of User Needs Stage constitute the design basis that subsequently supports and constrains the generation of target system solutions. For instance, conditions for selecting and synthesising particular generalised task model descriptions at the Composite Task Model Stage would be defined by SUN(y) and DoDD(y).

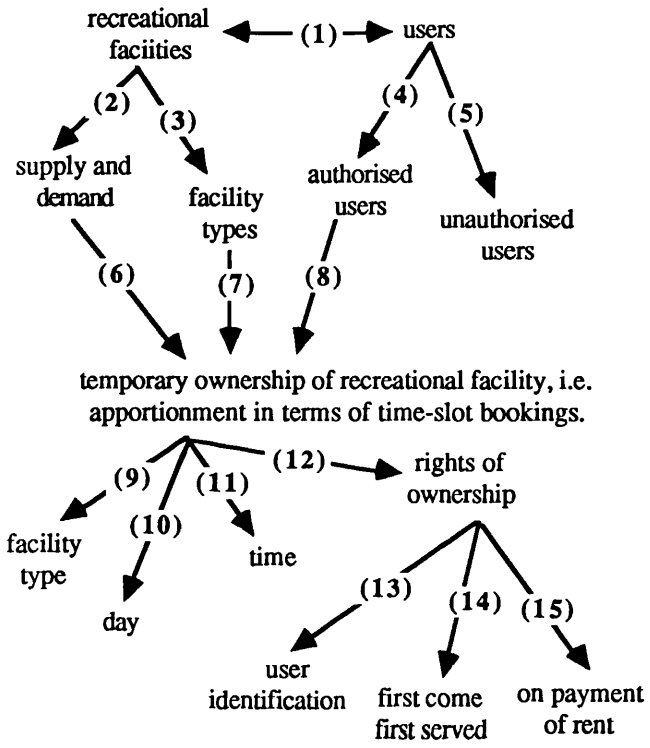
(IV) The Composite Task Model Stage

The objective of this stage is to establish a conceptual foundation for target system design. In particular, it defines the basis on which an appropriate functional design may be advanced. To this end, on-line and off-line tasks[8] are identified (this step

[7] The notation was adopted for convenience. Alternative notations may also be suitable.

[8] On-line and off-line tasks correspond to tasks which are supported and unsupported by the computer system respectively.

Figure AE-2 : Domain of Design Discourse Description for the Target Recreation Facility Booking System (DoDD(y))



No.	Description
(1) to (3)	The relationship between users and recreational facilities (1) is dictated by the supply and demand (2) of various types of recreation facilities (3).
(4) to (5)	There are two categories of users: authorised users (4) and unauthorised users (5).
(6) to (8)	Only authorised users (8) are allowed to use the facilities which are limited in supply (6) and type (7). A limited supply necessitates the temporal assignment of ownership.
(9) to (15)	To assign temporal ownership, one must define what constitutes: (a) a time slot (9 to 11), and (b) rights of ownership (12 to 15).

is termed *task* allocation) prior to a separation of the on-line task components into interactive and solely computer tasks (this step is commonly referred to as *function* allocation). These objectives are achieved as follows :

(a) a sub-set of GTM(x) is synthesised with GTM(y). The sub-set represents promising and compatible extant system(s) designs that satisfy the design basis defined at the Statement of User Needs Stage. The result of the synthesis is a preliminary composite task model of the target system termed CTM(y). To support stage-wise evaluation, iterative design, and post-implementation modifications and maintenance, additional notes on the composite task model are documented as shown in Table AE-1, e.g. insights associated with design features selected from GTM(x) and specific design constraints described by GTM(y);

(b) the preliminary composite task model is decomposed further to facilitate task allocation. Thus, a satisfactory level of description is derived iteratively with step (c) below. Note that the decomposition should comply with the design basis established at the Statement of User Needs Stage;
(c) on-line and off-line tasks are demarcated in the composite task model. Off-line tasks are indicated by grey envelopes (see Figure AE-3);
(d) interactive and automated tasks (i.e. solely computer tasks) are then decided in respect of the on-line task. The allocated functions are indicated on the structured diagram as H: and C: actions respectively (see Figure AE-6).

Figure AE-3 shows a CTM(y) description for the case-study. The shaded portion describes status checks to determine whether a user is authorised to make bookings. On positive verification, the user can then proceed to match recreation facility, day and time preferences with available booking slots. Booking processes for both successful and unsuccessful outcomes are thus described by CTM(y). On the basis of such a description, an appropriate task allocation may then be decided. In the context of the case-study, booking decisions are designated as off-line tasks since the selection criteria are too varied among users (i.e. largely indeterminate). These tasks are indicated by grey envelopes as shown in Figure AE-3.

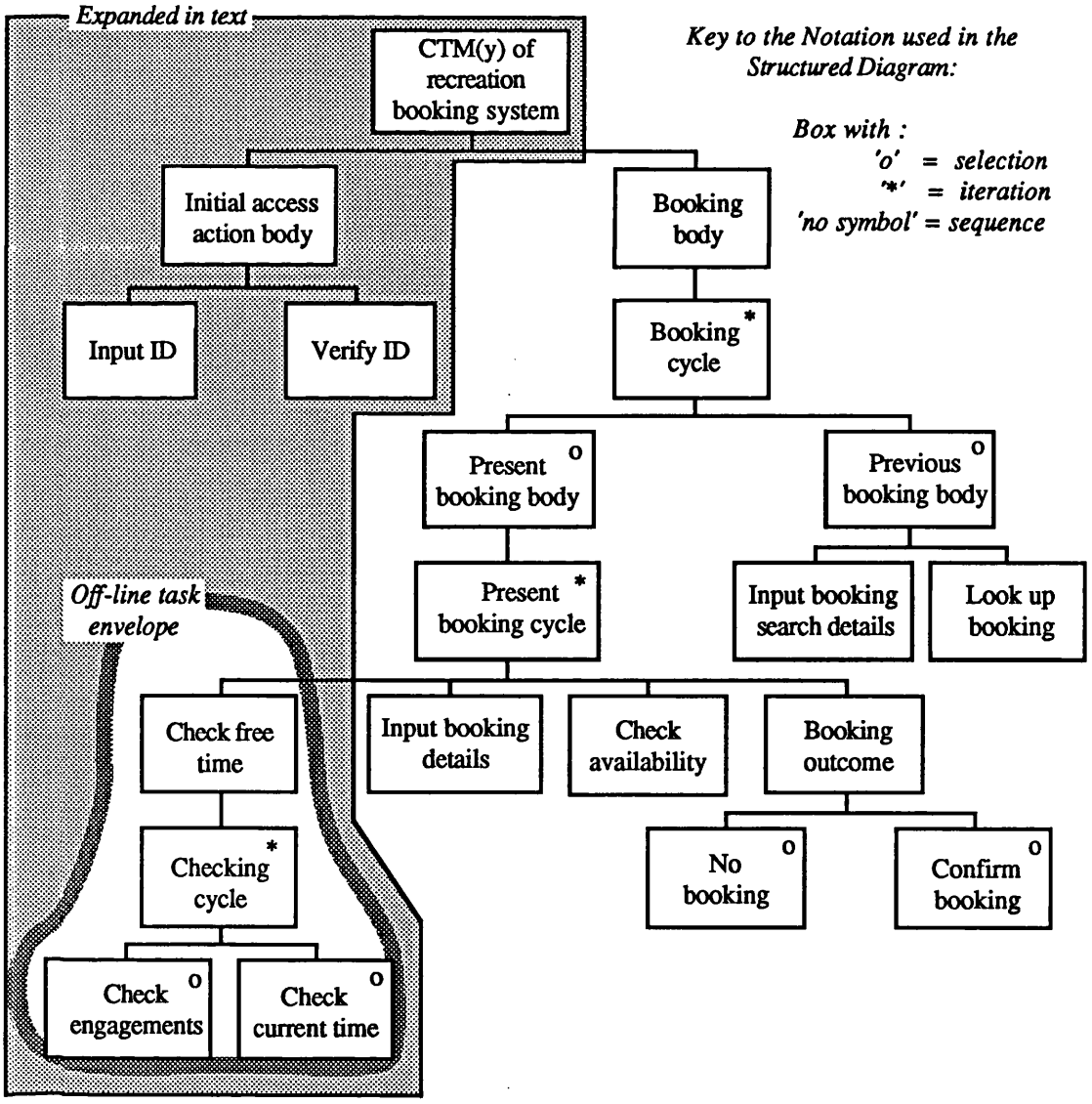
Two other aspects need to be highlighted in respect of the Composite Task Model Stage, namely :

(a) design iterations should be expected between the Statement of User Needs and Composite Task Model Stages. For instance, any modification to either SUN(y) or CTM(y) (e.g. arising from user feedback) would necessitate such iterations. In some instances, wider design implications may also be involved (see (b) overleaf). A case-study illustration follows. During initial design, it was assumed that user problems with the current system would be alleviated by displaying more booking information on a larger screen. Although this solution would improve system performance, the client may later decide on providing better computer support for checking previous reservations. Since computer-assisted search could

support both the task of checking present booking availability and previous reservations, previous design specifications concerning the provision of a richer information display on a larger screen would be superseded. Iteration around the Statement of User Needs, Composite Task Model, and System and User Task Model Stages (see later) is thus required to update their design descriptions;

(b) the first design inter-dependency between the streams of the JSD* method occurs at the Composite Task Model Stage (see Figure AE-4). The

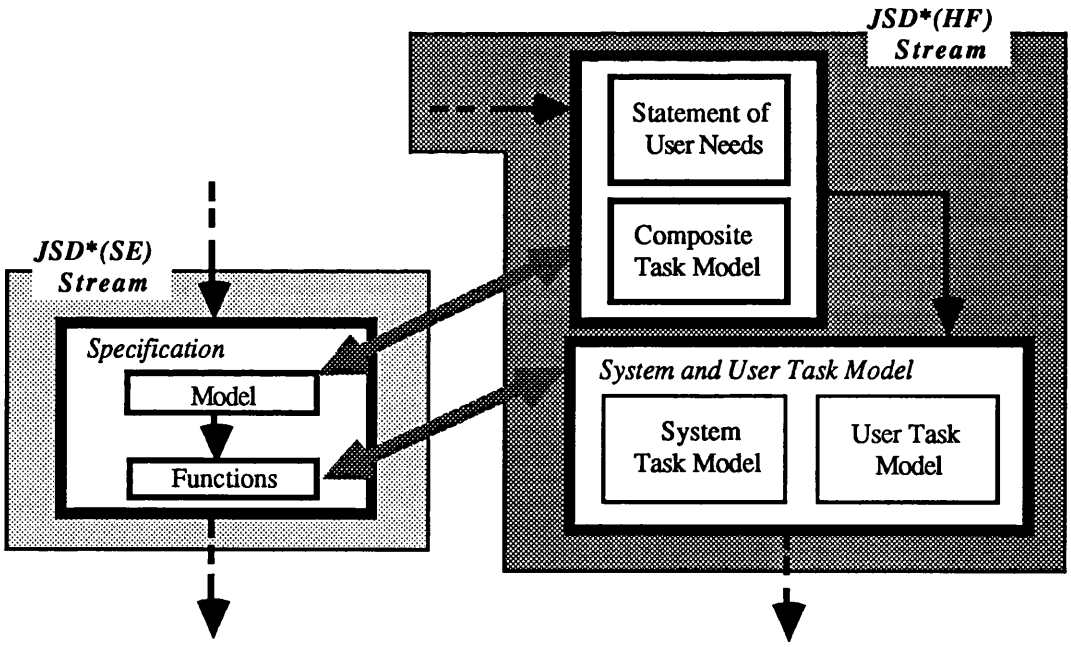
Figure AE-3 : Composite Task Model for the Target Recreation Facility Booking System (CTM(y))



inter-dependency intersects the latter JSD*(HF) stage with the Modelling Stage of the JSD*(SE) stream. At these stages, JSD*(SE) and JSD*(HF) design concerns are expected to overlap. Thus, design information of common interest should be shared. To this end, the information inter-dependency should, at a minimum, be determined by information requirements of the JSD*(SE) Modelling Stage. The minimum is set on the basis that information captured in the JSD*(HF) stream would not be relevant to JSD*(SE) design in its entirety. For instance, JSD*(SE) designers would not be interested in low level user requirements at this stage.[9]

Generally, products of the Composite Task Model, Statement of User Needs and Modelling Stages would be discussed between JSD*(HF) and

Figure AE-4 : Design Inter-dependencies between Software Engineering and Human Factors Streams of the JSD* Method



[9] This assertion follows because the JSD method does not explicitly address user requirements capture and documentation. In particular, the JSD Model is not concerned with user tasks (unlike JSD*(HF) products derived at this stage).

JSD*(SE) designers.[10] In particular, design information on object and action attributes, user needs and problems, system events, user task semantics and expected user task support, is shared and discussed. The result of the discussions is the specification of a *Functions List* to define the scope of target system design. The list is essentially a tabular description of the initiating trigger, end result and performance characteristics of functions to be supported by the system.

Presently, the case-study example described in (a) above is taken further to illustrate inter-dependency requirements between the JSD* design streams. In this case, the delayed introduction of additional functional support would not affect the JSD*(SE) model if an appropriate system scope was defined originally, i.e. actions and attributes of the modelled entity would remain unchanged. However, the additional functions would have to be accommodated at the Functions Stage of the JSD*(SE) stream. Thus, the original Functions List would have to be updated to account for the changes. In addition, a design iteration should be performed to update all JSD*(HF) products that are affected by the changes.

Having agreed on a Functions List, on-line and off-line tasks of the CTM(y) description are updated accordingly by JSD*(HF) designers. The description is then decomposed further at the next stage of the method.

(V) The System and User Task Model Stage

Three design concerns are addressed at this JSD*(HF) stage, namely :

(a) on-line and off-line components of CTM(y) are decomposed further to support function allocation. Bearing in mind appropriate contributions from extant systems analysis and the JSD*(SE) stream (see (b) and (c) below),

[10] A wider scope of design discussions and information sharing is not precluded at this inter-dependency point. Indeed, wider discussions may be undertaken to pre-empt the design information requirements of the JSD*(SE) Functions Stage.

on-line components are decomposed to yield the system task model of the target system (i.e. STM(y)). In other words, the system task model re-describes the on-line task in terms of lower level human and computer tasks. Functional design of the target system is then pursued via the system task model.

Meanwhile, off-line components of the composite task model are collated into a single description termed the user task model of the target system (i.e. UTM(y)). In other words, the model comprises a summary of tasks that are unsupported by the computer. In many cases, the user task model is not decomposed further (see later).

Figure AE-5 illustrates graphically how the CTM(y) description for the case-study is decomposed and documented using JSD* structured diagram notation. The primary product of the decomposition, namely STM(y), is detailed in Figure AE-6. This Figure shows that the STM(y) description essentially comprises high level cycles of human-computer interaction, e.g. computer prompts and user inputs.

As regards off-line tasks, it was decided that further decomposition would not be necessary since the user interface design for the present case-study, would not be influenced significantly by such tasks. Consequently, off-line tasks were collated directly into a UTM(y) description (see Figure AE-5). In other cases, UTM(y) may be decomposed further to support job design, e.g. when both STM(y) and UTM(y) descriptions have to be assessed to determine whether the combined workload is acceptable;

(b) suitable sub-sets of extant system descriptions (specifically STM(x) and UTM(x)) are selected and incorporated in accordance with GTM(x) and SUN(y). For the case-study, it was decided that two booking scenarios from the current system would be accommodated. The scenarios characterise the requirements of the following user groups :

(i) users whose time for recreation is tightly constrained;

(ii) users whose time for recreation is flexible. For such users, booking preference is concerned only with recreation type and day.

These user characteristics should be accommodated when the recruitment and enhancement of extant designs are considered. As an illustration, consider the design of search-and-match functions to support recreation booking. To accommodate the requirements of (i) above, the search function would have to support the location of specific booking slots. Thus, the design is required to prompt the user for a complete input of booking

Figure AE-5 : Decomposition of CTM(y) into STM(y) and UTM(y)

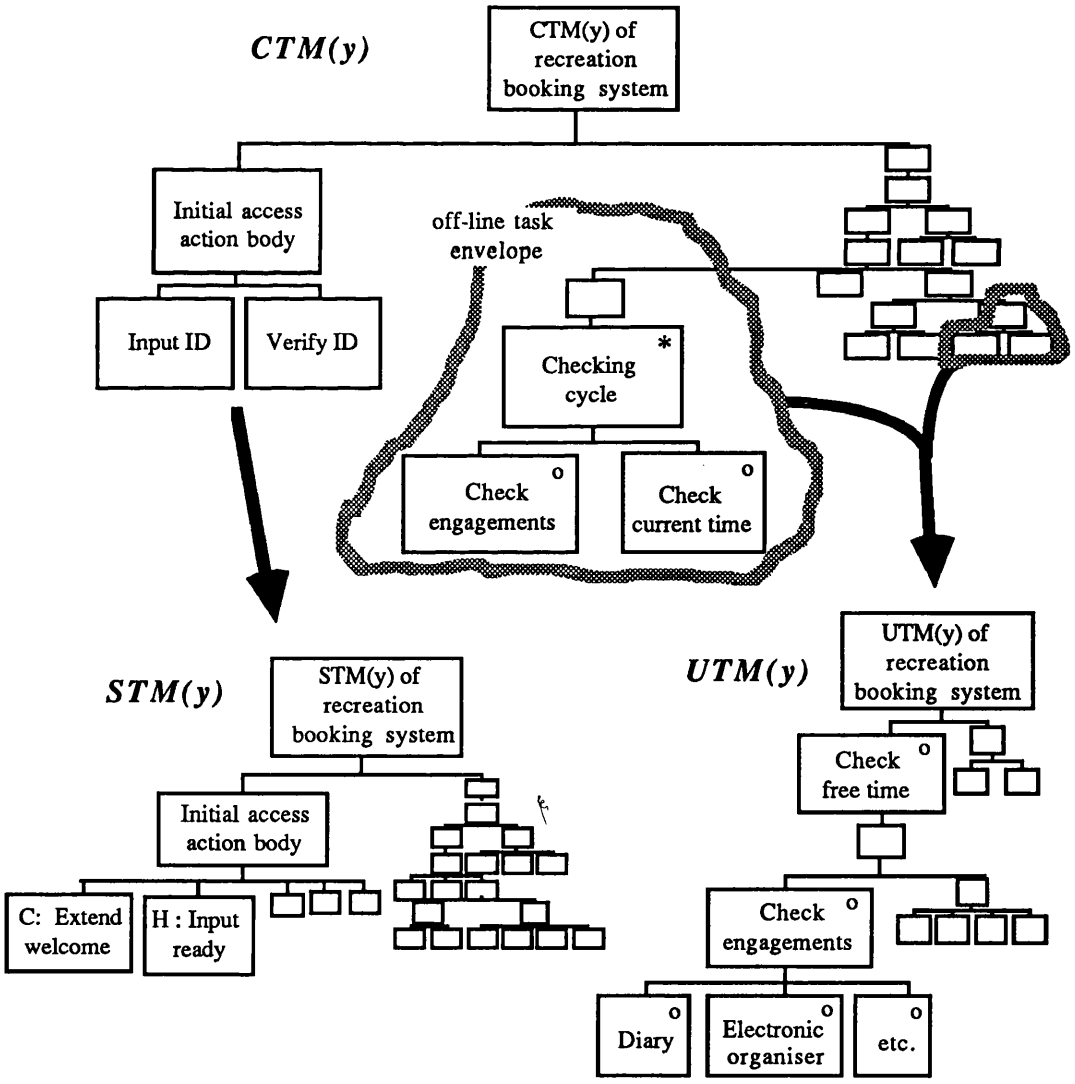
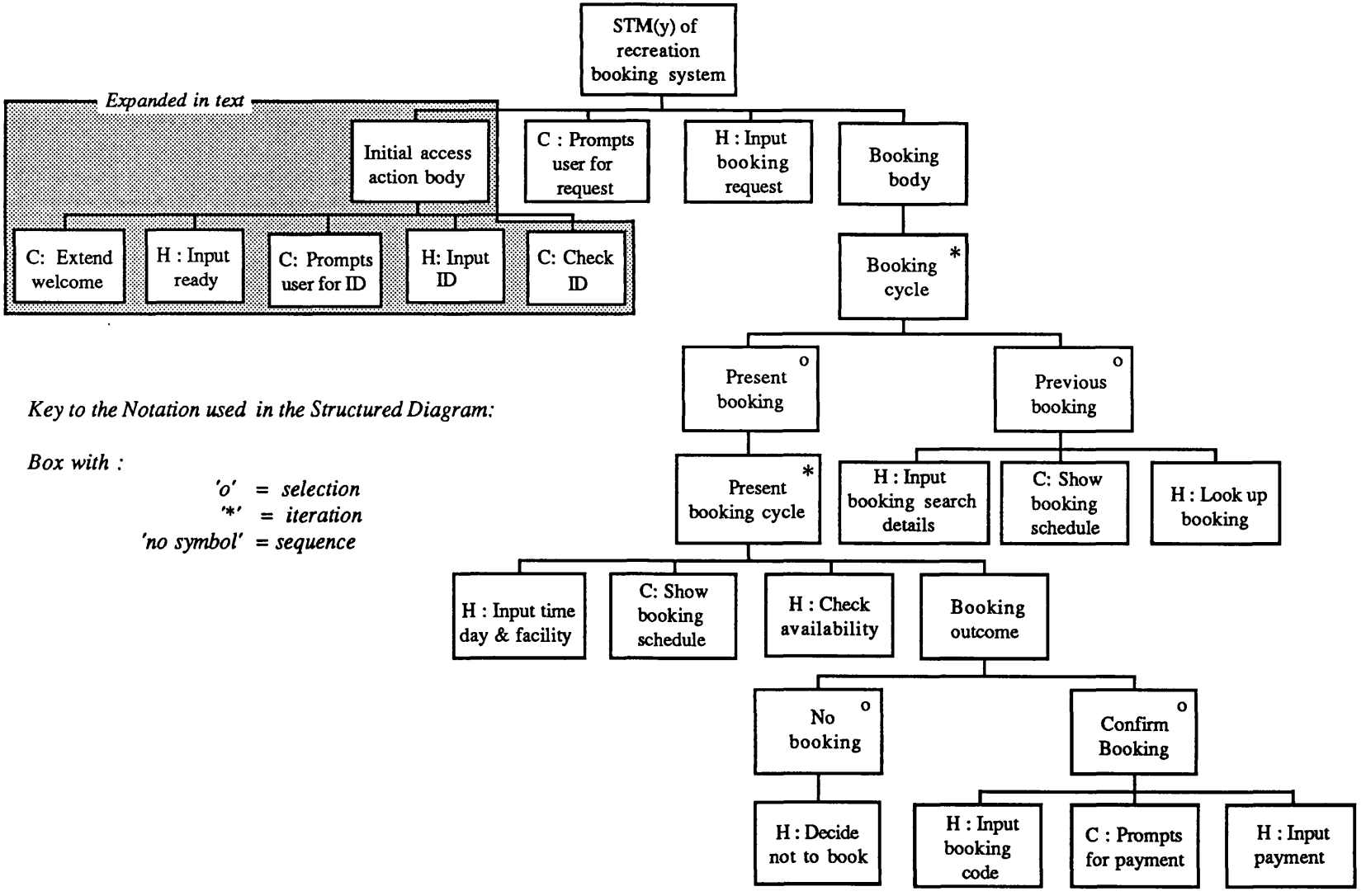


Figure AE-6 : System Task Model for the Target Recreation Facility Booking System (STM(y))



Key to the Notation used in the Structured Diagram:

Box with :

'o' = *selection*
''* = *iteration*
'no symbol' = *sequence*

parameters (comprising time band, recreation type, and day of desired booking). As for the requirements of (ii) above, the search function would have to support the checking of booking availability by recreation type and day.

Following the above considerations, it may be concluded that current system designs which prompt disparate user input of recreation type and day (i.e. one input per interaction cycle) are irrelevant to target system design. An explanation of the conclusion follows. Disparate input was assumed for the current system to minimise the information that has to be displayed on each screen. The latter constraint was imposed by the client's selection of an eight-line LCD screen. To accommodate the constraint (on the current system) without lowering unacceptably the chances of offering suitable booking slots to the user, booking parameters had to be organised sequentially to support a stage-wise convergence scheme. Since the hardware constraints no longer apply for the target system, more booking information may be displayed on each screen to support multiple user selections. Thus, a multi-variate search function may be introduced. In other words, users may specify conjunctions of booking parameters to tightly constrain the booking information retrieved by the computer. Consequently, current system designs that support stage-wise convergence are no longer relevant. However, other extant designs such as booking input and confirmation prompts may still be ported to the target system (if appropriate);

(c) design information of common interest is shared at this inter-dependency with the Functions Stage of JSD*(SE) stream (see Figure AE-4). At this stage, both streams of the JSD* method are concerned largely with functional decomposition. As such, the functions list (defined at the previous inter-dependency) would not be specific enough to ensure that a convergent design is derived by both JSD* streams. Thus, more specific design information has to be shared and agreed between JSD*(HF) and JSD*(SE) designers. In particular, the design information to be shared

would comprise the following :

(i) JSD*(HF) design information -- STM(y) description of human-computer interaction cycles and sequencing of functional supports;
(ii) JSD*(SE) design information -- descriptions of input and output streams, JSD function and model processes, and the input sub-system (contributes to the design of error and feedback messages at later stages of the JSD*(HF) method).

Following discussions on the above design descriptions, a common pool of information is adopted to ensure design convergence. In addition, close contact should be maintained between JSD* streams to avoid unnecessary design iterations when JSD*(HF) and JSD*(SE) specifications are integrated.

For a case-study of this design inter-dependency, the reader is referred to Section IV where an example concerning the checking of previous bookings was described.

The scope of the System and User Task Model Stage may now be summarised. A STM(y) description is derived to define the high level human-computer interactions required by the on-line task. Taking due account of the design basis defined at the Statement of User Needs Stage, device level design is then pursued via STM(y). In this respect, pertinent characteristics of UTM(y) should also be noted (e.g. information exchanges with STM(y)) since they could contribute to job and user interface design, e.g. the content and format of information displays; the requirement for speech input devices arising from off-line task demands; combined workload assessments; etc.

(VI) The Interaction Task Model Stage

The objective of this stage is to specify the device level inputs implicated by the interactive task. In particular, the product of this stage, namely ITM(y), is an

'idealised' description of the user inputs required to advance the interactive task. On the basis of such a description, potential user errors are then considered at a later design stage (refer to Section VIII).

ITM(y) is derived by decomposing the H: leaves of the STM(y) description (corresponding to user actions -- see Figure AE-7). The resulting description is expressed in terms of the following :

(a) object and action primitives of the chosen user interface environment (if any);
(b) basic keystrokes of the designated hardware.

ITM(y) should be described at a level easily understood by design team members and end-users. For instance, the ITM(y) description for the case-study is terminated at the level of mouse clicks (instead of further decomposition into a mouse-down followed by a mouse-up -- see Figure AE-8), since the Macintosh interface style was assumed to be understood by all concerned.

As with earlier JSD*(HF) products, ITM(y) is documented using the JSD* structured diagram notation.

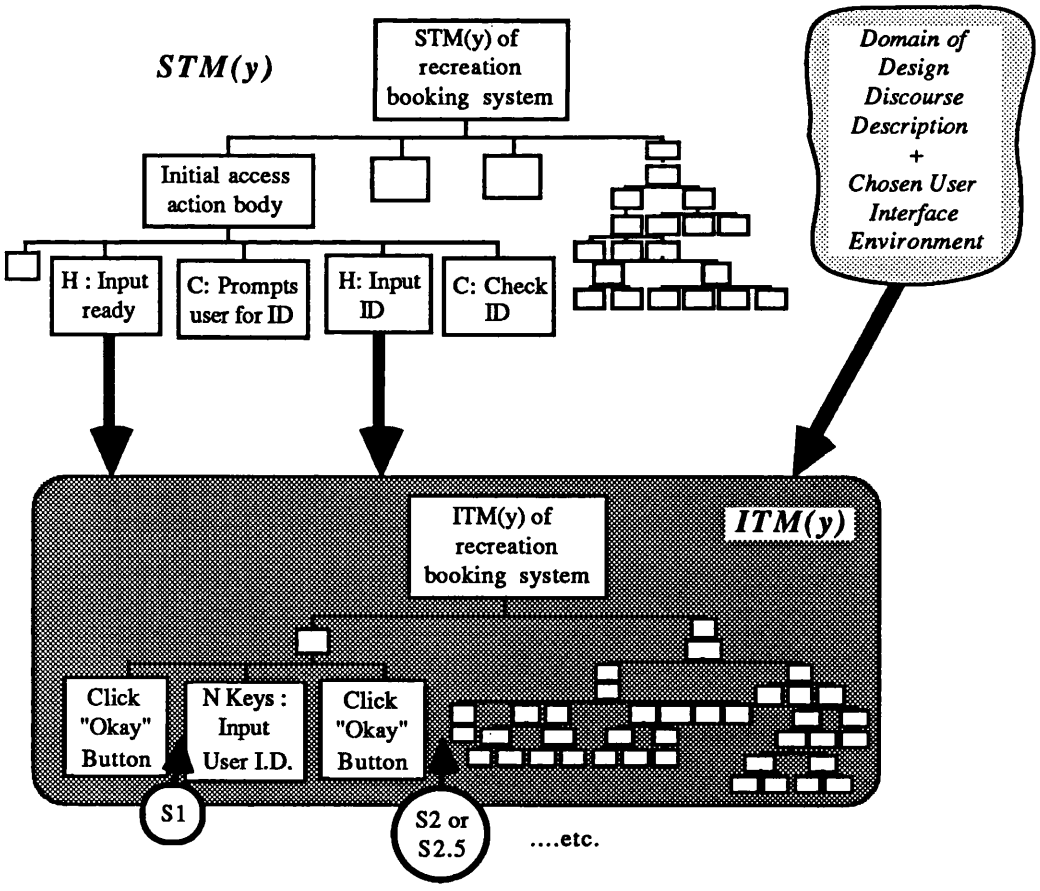
To derive an appropriate level of ITM(y) description, design iterations *within* the stage and *across* succeeding stages may be necessary. An appropriate description is required to support later grouping of ITM(y) leaves into coherent or meaningful sub-tasks. Each of these groups are annotated on ITM(y) as a 'bubble' containing a screen number, e.g. Screen 1 or S1 in Figure AE-8. The annotation constitutes part of an overall scheme to *inter-link* products of the *Design Specification Phase* of the JSD*(HF) method (see Figure 8-1). Specifically, the ITM(y) description is linked with products of the Display Design Stage. Thus, user inputs are linked with dynamic screen actuations (to present functional supports, and error and help messages to the user -- see later) in accordance with the interactive task context.

A case-study illustration of the inter-linkages follows. An inspection of Figures

AE-8 and AE-11 reveals that the bubble labelled 'S2.5' (or Screen 2.5) appears in both ITM(y) and the dialogue and inter-task screen actuation description (refer to Section VIII for a detailed account). The appearance of Screen 2.5 across these products implies the following scenario : an illegal input is detected by the computer when an unauthorised user enters an invalid identification number and clicks on the 'Okay' button. Thus, screen 2.5 with error message 2 (denoted as 'Screen 2.5 -- em 2' in Figure AE-11) is displayed by the computer.

To complete the specifications for the scenario, the content of error message 2 or em 2 is described in the message index (Figure AE-10) as 'Invalid personal i.d.'

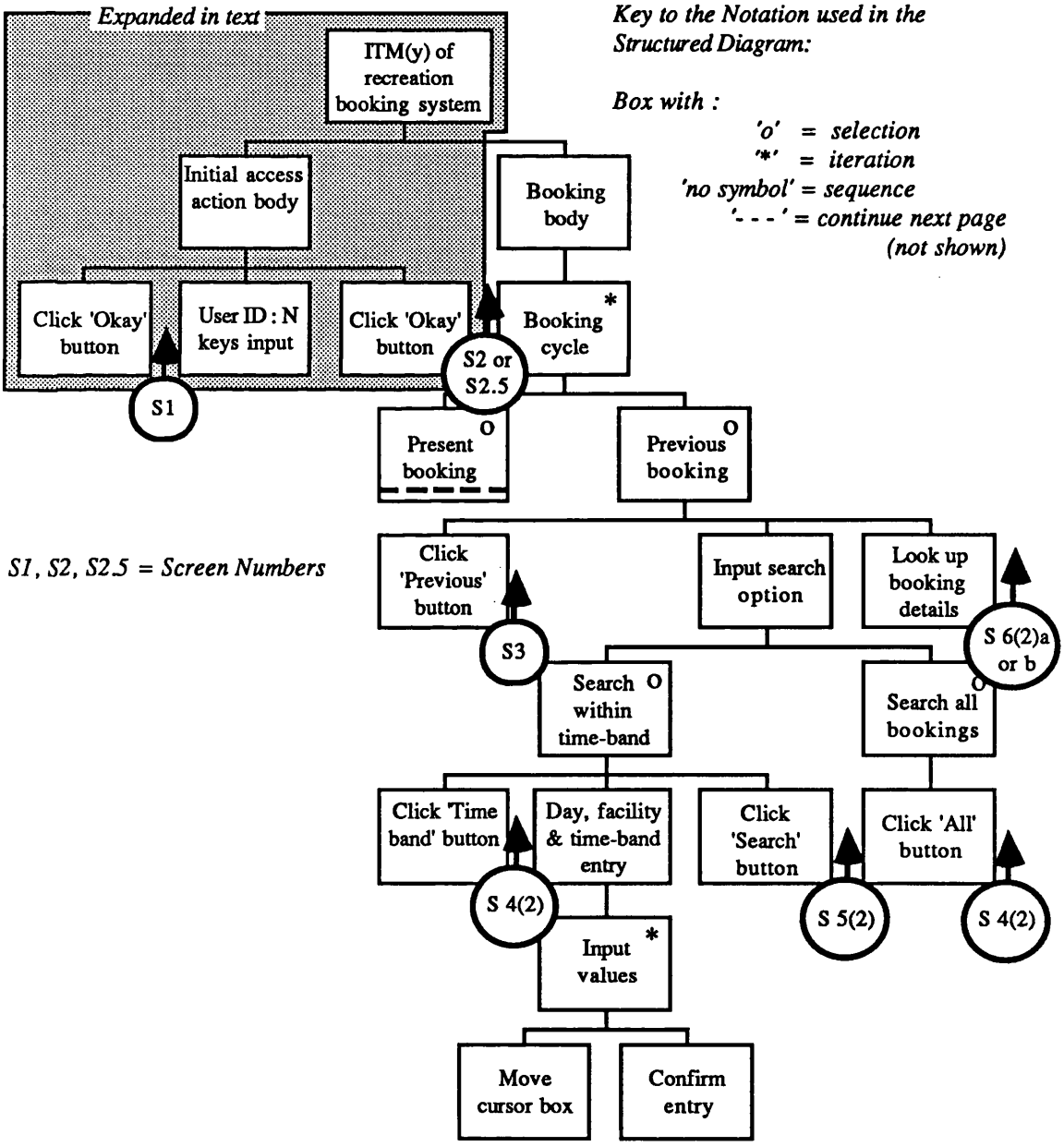
Figure AE-7 : Deriving ITM(y) by Decomposing Human (H:) Leaves of STM(y) on the Basis of the Domain of Design Discourse Description and the Chosen User Interface Environment



S1, S2, S2.5 = Screen Numbers

Similarly, the design of screen 2.5 is determined by referring to another JSD*(HF) product which describes its composition and layout pictorially (see Figure AE-12 for a description of Screen 1). Finally, the behaviours of screen components are specified in a set of JSD*(HF) descriptions termed the interface model or IM(y) descriptions. These descriptions are discussed in greater detail in the next Section.

Figure AE-8 : Interaction Task Model for the Target Recreation Facility Booking System (ITM(y))



(VII) The Interface Model Stage

The objective of this stage is to specify the behaviour and changes in appearance (if any) of *bespoke* screen objects, e.g. object responses to user inputs and changes in states of representation and real world entities. Screen objects originating from the chosen user interface environment (if any) need not be described since they may be assumed to be understood by design team members. The product of this stage is a set of interface model descriptions or namely IM(y).

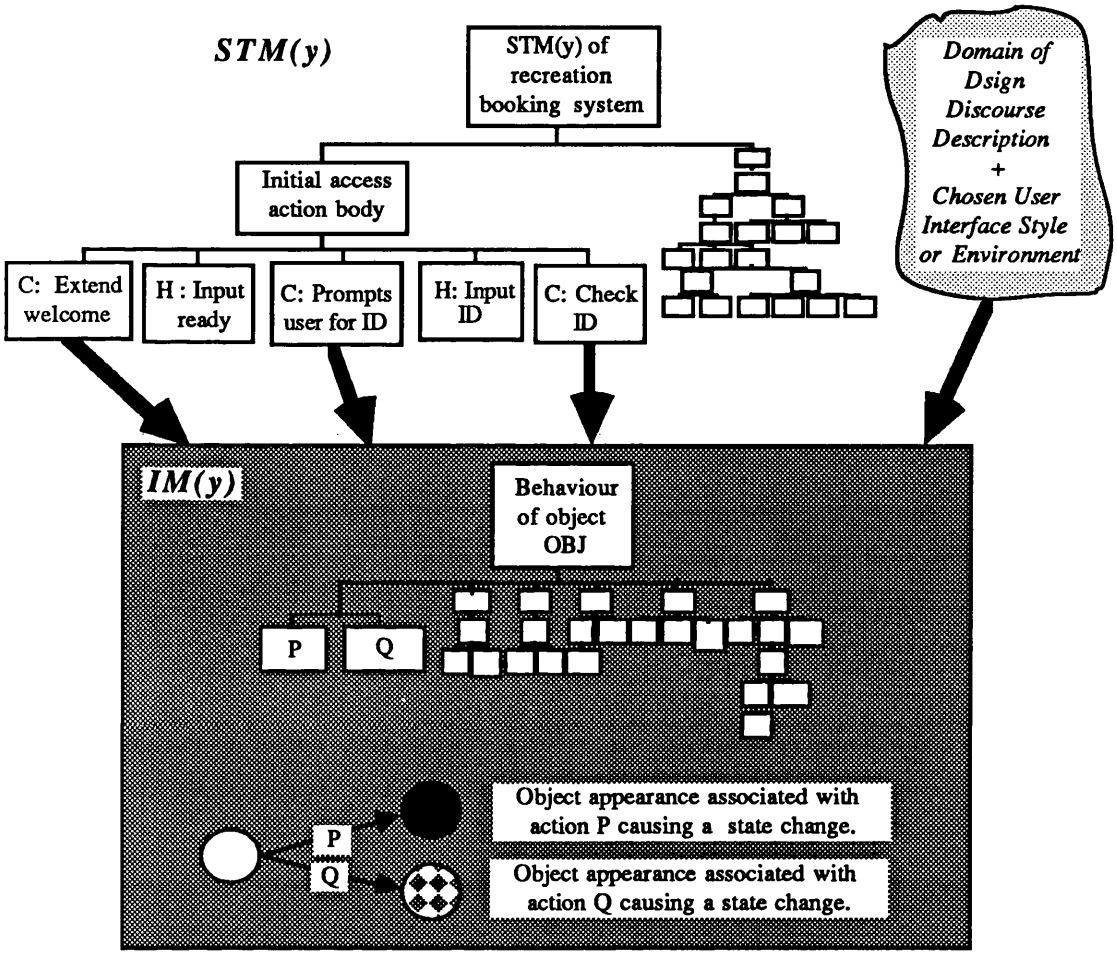
IM(y) descriptions are derived by decomposing the C: leaves (i.e. computer actions) of STM(y) (see Figure AE-9). The decomposition should be consistent with the context established by the domain of design discourse description. Contextual consistency also applies when a secondary conceptual framework (such as a user interface metaphor) and/or a particular user interface environment is adopted, e.g. the adopted metaphor should be compatible with the system domain. To support later assessment of such concerns, the design decisions and rationale should be documented.

A case-study example of a structured diagram description of IM(y) is shown in Figure AE-9. The Figure illustrates how changes in the states of an abstract screen object OBJ (attributed to actions P and Q) may be linked to its appearance changes (pictorial descriptions). Thus, icon design falls within the scope of the Interface Model Stage.

IM(y) descriptions are carried forward to the Display Design Stage to support screen composition. Links with the dialogue and inter-task screen actuation description may also occur since action leaves of particular IM(y) descriptions constitute triggers for screen actuations. A case-study example of such a link was described in Section VI, i.e. a mouse click on the 'Okay' button object was linked to the actuation of Screen 2.5.

Presently, the products derived at the Display Design Stage are described.

Figure AE-9 : Deriving IM(y) by Decomposing Computer (C:) Leaves of STM(y) on the Basis of the Domain of Design Discourse Description and the Chosen User Interface Environment



(VIII) The Display Design Stage

At this stage, preceding JSD* design descriptions are drawn together to complete the specification of a software user interface design. In particular, the JSD*(HF) stream contributes IM(y) and ITM(y) descriptions, while specifications of the JSD input sub-system and output contents of information function processes are contributed by the JSD*(SE) stream. On the basis of these descriptions, the following specifications are addressed :

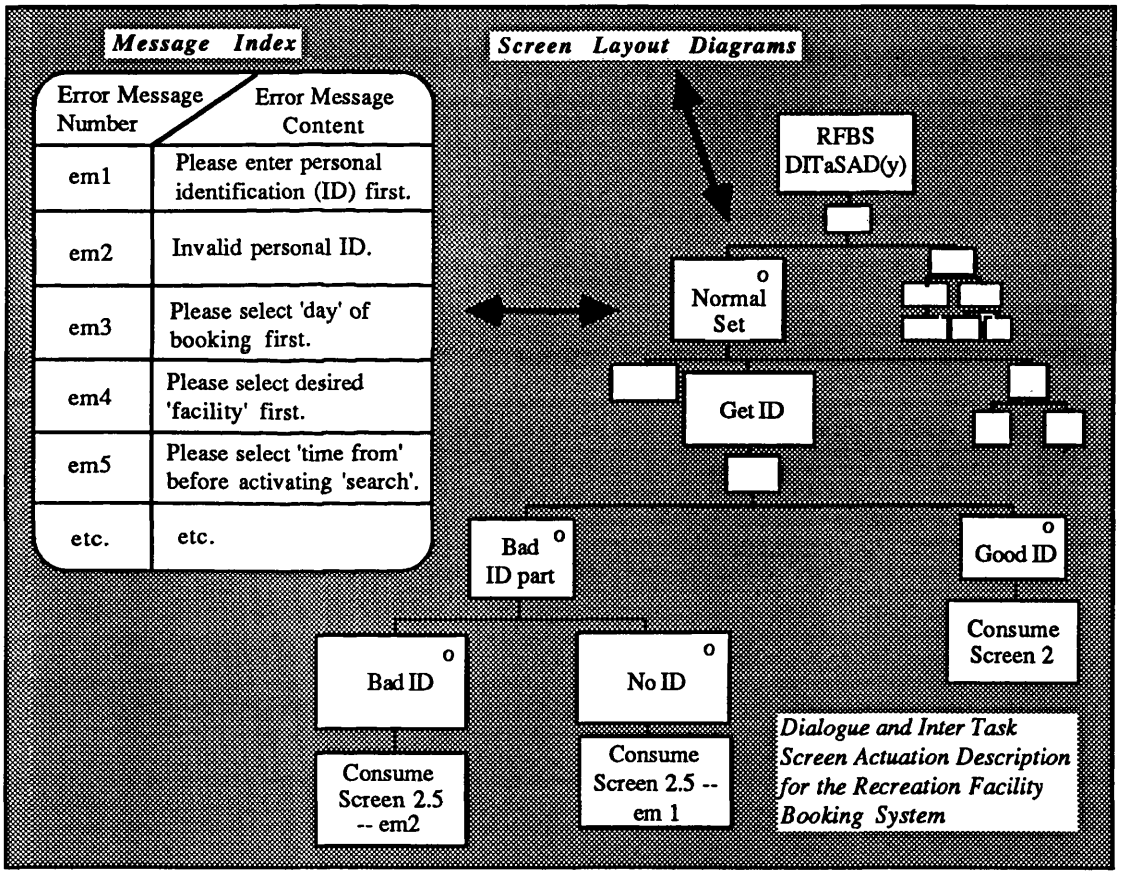
(a) the content of error, feedback and help messages. These specifications

are described by a message index (Figure AE-10);

(b) the context for triggering error, feedback and help messages. These specifications are described by a dialogue and inter-task screen actuation description (Figure AE-11);

(c) the composition and layout of screen displays. These specifications are described by screen layout diagrams[11] (Figure AE-12). A dictionary is also

Figure AE-10 : Inter-Linkages between Products of the Display Design Stage

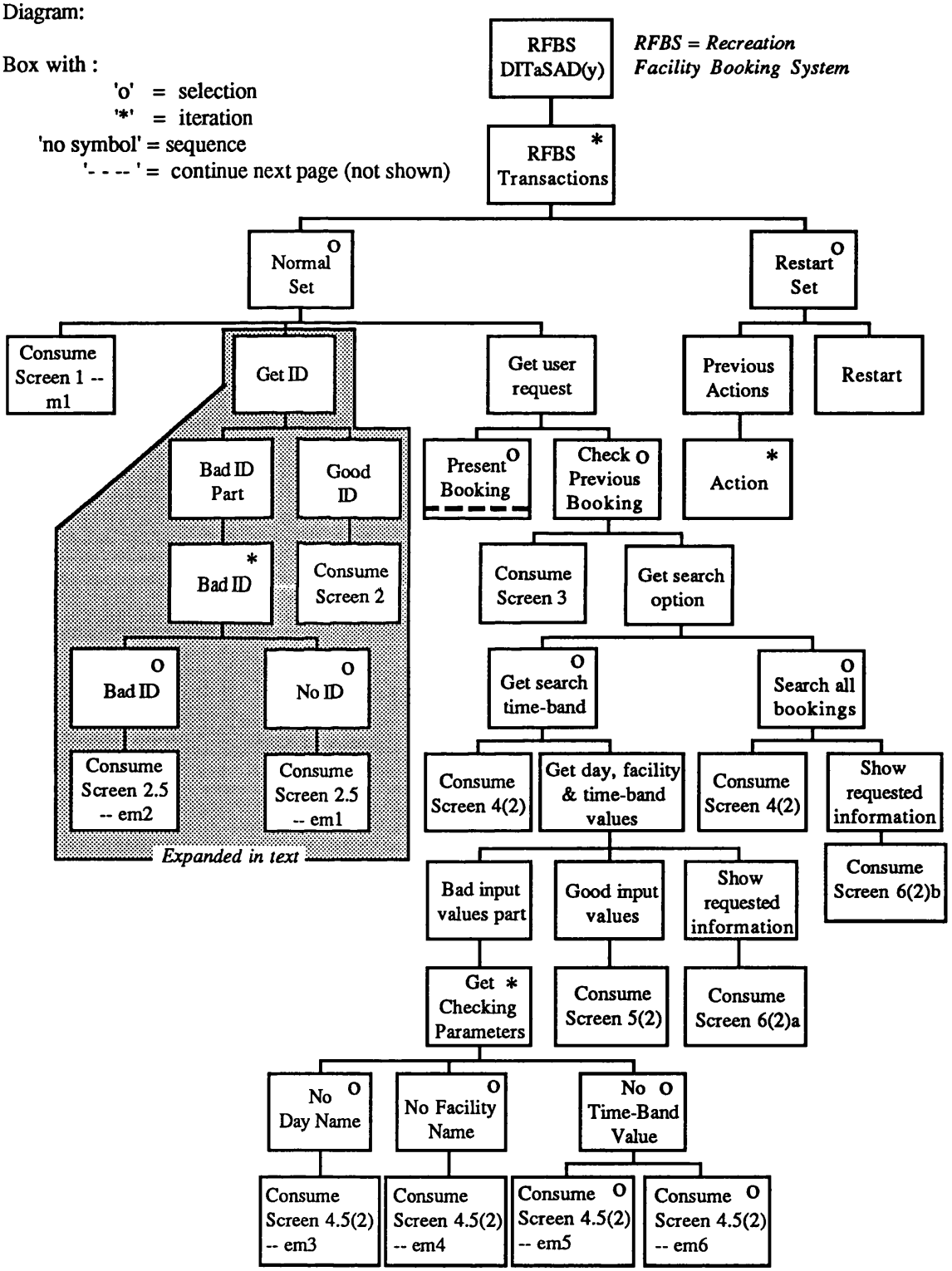


[11] Screen layout diagrams may be drawn on paper (scale or dimensioned drawings) or prototyped using a computer-based tool (e.g. Prototyper™). Although paper-based documentation of the message index and screen layout diagrams may be superseded by such tools, it is emphasised that the dictionary of screen objects, interface model descriptions, and the dialogue and inter-task screen actuation description should still be documented explicitly. Thus, the requirement for comprehensive documentation to support design evaluation and maintenance is satisfied.

Figure AE-11 : Dialogue and Inter-Task Screen Actuation Description for the Target Recreation Facility Booking System (DITaSAD(y))

Key to the Notation used in the Structured Diagram:

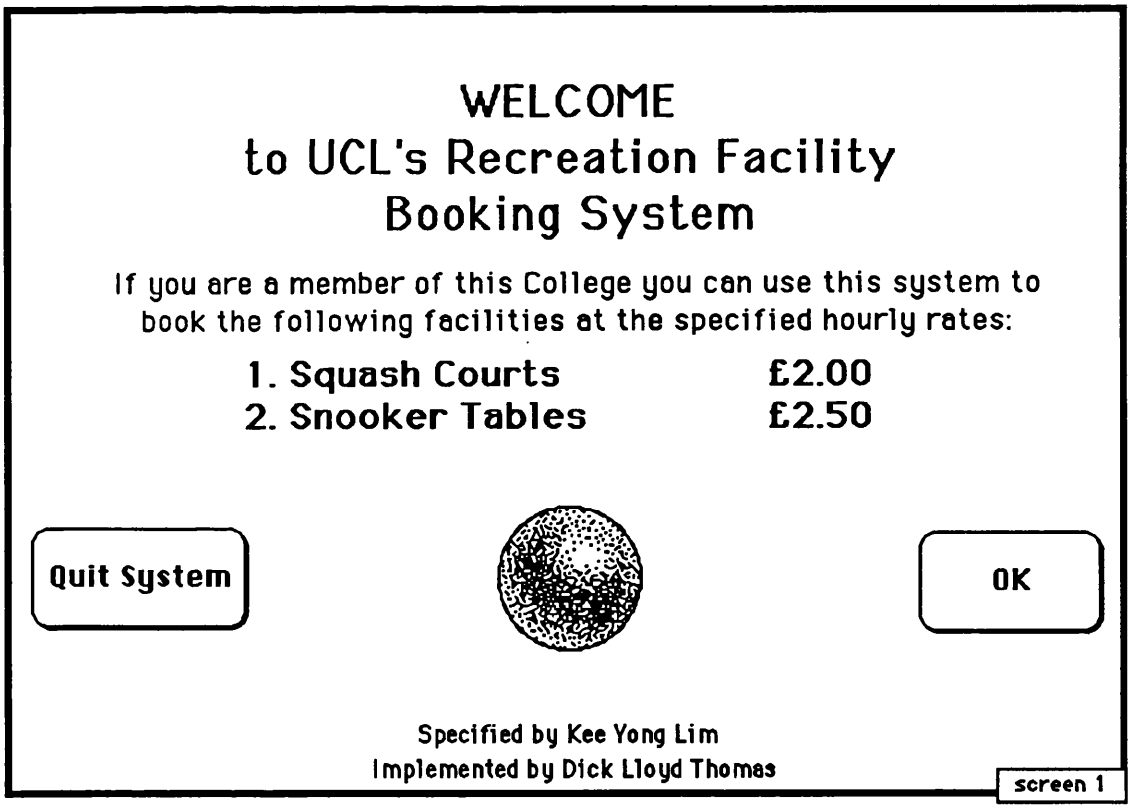
Box with :
'o' = selection
'*' = iteration
'no symbol' = sequence
'- - -- ' = continue next page (not shown)



included to provide a textual account of objects in each screen diagram (Table AE-2).

The Dialogue and Inter-Task Screen Actuation Description (DITaSAD(y)) describes how screen actuations are set against the interactive task context described by ITM(y). In other words, it specifies when particular computer functions and messages should be presented to support interactive task performance.[12] Wider linkages between DITaSAD(y) and other JSD*(HF) products (namely interface model descriptions, message index and screen layout diagrams) have already been described in Section VI.

Figure AE-12 : A Pictorial Screen Layout Description for the Target Recreation Facility Booking System



[12] The objective of DITaSAD(y) is not the description of all possible screen transitions. For instance, transient changes in the appearances of objects (these are described by IM(y)), screen refresh and scrolling actions, etc., are excluded from its scope.

Table AE-2 : Dictionary of Screen Objects for the Target Recreation Facility Booking System (Screen 1)

Screen Object	Description	Design Attributes
Okay button	The user clicks this button to indicate that the welcome message and membership requirements are understood.	On input, the computer displays Screen 2 to prompt the user to input an identification number.
.....etc.	.....etc.	.....etc.

To summarise, structured diagrams, tables and pictorial diagrams are used to document the dialogue and inter-task screen actuation description, message index and screen layout respectively. These products of the Display Design Stage, together with ITM(y) and IM(y), comprise JSD*(HF) specifications of a user interface design.[13] The specifications are discussed with JSD*(SE) analysts.[14] Amalgamated JSD*(HF) and JSD*(SE) specifications are then implemented in

[13] The potential for wider applications of JSD*(HF) products should be noted. For instance, products derived between the Generalised Task Model and Interaction Task Model Stages (inclusive) could support the design of training programmes and user manuals. Specific applications of each JSD*(HF) product have already been indicated in the paper, e.g. job design would be supported by system and user task model descriptions. Further account of the relationships between JSD*(HF) products and existing human factors design 'topics' is excluded, since the main objective of the present research is the specification of JSD*(HF) descriptions to support system design.

[14] In view of the current training of human factors designers and the notation adopted by the method, the amalgamation of JSD*(HF) with JSD*(SE) specifications should be undertaken by JSD analysts. The assertion is consistent with the design roles currently assumed by human factors designers and software engineers.

accordance with the JSD method.[15, 16] Late evaluation[17] follows and 'final' design modifications (if any) are accommodated accordingly.

The above account completes a stage-wise illustration of the JSD*(HF) method.

To conclude, the JSD*(HF) method contributes to system development as follows :

(a) it specifies an explicitly structured human factors design process. Thus, the scope of human factors support is extended throughout the system design cycle. In addition, a structured design conception would support better project management since it encourages an explicit accommodation of human factors by the design agenda;
(b) it identifies explicit intersections between Human Factors and Software Engineering design analysis and specification. Thus, early, timely and contextually relevant human factors input is facilitated.

---

[15] Computer-based support for implementing JSD*(HF) specifications would be desirable.

[16] A detailed account of human factors input to JSD implementation is excluded from the present research. However, it may be expected that human factors input would be confined largely to the specification of additional feedback displays to accommodate particular JSD implementations. For instance, additional feedback would be necessary if an implementation involves longer than expected transient response times. Wider human factors input is not envisaged because the transformation of JSD* specifications implicated by JSD implementation, is a well regulated and mechanistic process. In addition, the transformations, by definition, would not alter the external behaviour implied by the specifications (see Zave, 1984).

[17] Late evaluation was excluded from the present research for the following reasons :

(a) human factors input to late evaluation is already well established. Indeed, off-the-shelf evaluation techniques are readily available for recruitment to the JSD*(HF) method;
(b) the primary objective of the research is to address the *'too-little-too-late'* problem of human factors input to system development. Thus, the focus of the research is on the development of human factors support for early as opposed to later stages of system design, i.e. to support design *analysis* and *specification*, rather than design *implementation* and *evaluation*.

Bibliography

(A) List of Published Papers Originating from the Present Research

Lim, K. Y. and Long, J. B., 1992, A Method for (Recruiting) Methods : Facilitating Human Factors Input to System Design. In : Brooks, R. (ed.), Proceedings of the ACM Annual Conference on Human Factors in Computing Systems (CHI'92), Monterey (USA), May 3-7, 1992, ACM.

Lim, K. Y. and Long, J. B., 1992, Computer-Based Tools for a Structured Human Factors Method. In : Mattila, M. (ed.), Proceedings of the International Conference on Computer-Aided Ergonomics and Safety (CAES'92), Tampere (Finland), May 18-20, 1992, Elsevier Science Holland.

Lim, K. Y. and Long, J. B., 1992, Pitfalls of Rapid Prototyping : Observations on a Commercial System Development Project. In : Lovesey, E. J. (ed.), Contemporary Ergonomics, Proceedings of the Ergonomics Society's 1992 Conference, Aston (UK), April 7-10, 1992, Taylor and Francis.

Lim, K. Y. and Long, J. B., 1992, Rapid Prototyping, Structured Methods and the Incorporation of Human Factors in System Design. To appear in : MacLean, A. (ed.), The St. Petersburg International Workshop on Human Computer Interaction, Int. Centre of Scientific and Technical Information (Moscow), ACM SIGCHI (USA) and HFS (USA), August 4-7, 1992, St. Petersburg).

Lim, K. Y. and Long, J. B., 1992, Instantiation of Task Analysis in a Structured Method for User Interface Design. To appear in : Wearn, Y. (ed.), Task Analysis in HCI Workshop, European Association for Cognitive Ergonomics (EACE) and US National Centre for Geographic Information and Analysis (NCGIA), Schaerding (Austria), June 9-11, 1992.

Lim, K. Y., Long, J. B. and Silcock, N., 1992, Integrating Human Factors with System Development : An Overview of a Structured Human Factors Method. In : Hitchins, D. K. (ed.), Proceedings of the International Conference on Information-Decision-Action Systems in Complex

Organisations (IDASCO'92), Oxford (UK), April 6-8, 1992, IEE.

Lim, K. Y., Long, J. B. and Silcock, N., 1992, Integrating Human Factors with the Jackson System Development Method : An Illustrated Overview. In : Barber, P. and Laws, J. (eds.), Ergonomics (Special Issue on Cognitive Ergonomics III), 1992, 33 (12), Taylor & Francis, London.

Lim, K. Y., Silcock, N. and Long, J. B., 1991, Case-Study Illustration of a Structured Method for User Interface Design. In : Lovesey, E.J. (ed.), Contemporary Ergonomics, Proceedings of the Ergonomics Society's 1991 Conference, Southampton, April 1991, Taylor and Francis.

Lim, K. Y., Long, J. B. and Silcock, N., 1990, Motivation, Research Management and a Conception for Structured Integration of Human Factors with System Development Methods: An Illustration Using the Jackson System Development Method. In : van der Veer, G et al (eds.), Proceedings of the Fifth European Conference on Cognitive Ergonomics, Urbino (Italy), September 3-6, 1990, 359-374, Golem Press.

Lim, K. Y., Long, J. B. and Silcock, N., 1990, Integrating Human Factors with Structured Analysis and Design Methods: An Enhanced Conception of the Extended Jackson System Development Method. In : Diaper, D. et al (eds.), Proceedings of the Third IFIP Conference on HCI (Interact '90), 225-230, Elsevier Science Publishers, North Holland.

Lim, K. Y., Long, J. B. and Silcock, N., 1990, Requirements, Research and Strategy for Integrating Human Factors with Structured Analysis and Design Methods: The Case of the Jackson System Development Method. In: Lovesey, E.J. (ed.), Contemporary Ergonomics, Proceedings of the Ergonomics Society's 1990 Conference, Leeds, April 1990, 32-38, Taylor and Francis.

Silcock, N., Lim, K. Y. and Long, J. B., 1991, A Structured Method for User Interface Design. In : Queinnec, Y. and Daneillou, F. (eds.), Proceedings of the 11th Congress of the International Ergonomics Association's 1991 Conference, Paris, July 1991, 67-68, Taylor and Francis.

Silcock, N., Lim, K. Y. and Long, J. B., 1990, Requirements and Suggestions for a Structured Analysis and Design (Human Factors) Method to Support the Integration of Human Factors with System Development. In : Lovesey, E.J. (ed.), Contemporary Ergonomics, Proceedings of the Ergonomics

Society's 1990 Conference, Leeds, April 1990, 425-430, Taylor and Francis.

Walsh, P., Lim, K. Y., Long, J. B. and Carver, M. K., 1989, JSD and the Design of User Interface Software. In : Barber, P. and Laws, J. (eds.), Ergonomics (Special Issue on Methodological Issues in Cognitive Ergonomics), Vol 32, No 11, November 1989, 1483-1498, Taylor & Francis London.

Walsh, P., Lim, K. Y., Long, J. B. and Carver, M. K., 1988, Integrating Human Factors with System Development. In : Heaton, N. & Sinclair, M. (Eds.) Designing End-User Interfaces. Oxford : Pergamon Infotech State of the Art Reports 15:8, 1988, 111-120, Pergamon Infotech Limited England.

(B) Cumulative List of RARDE Project Working Documents		Author(s)
WD 1	Integrating Human Factors into System Development -- The Project Proposal	K. Y. Lim
WD 2	Elaboration of the Aims stated in the Project Planning Document # 1	P. Walsh & K. Y. Lim
WD 3	Structured Methods and the Design of Interactive Software	P. Walsh & K. Y. Lim
WD 4	Formal Methods of Software Development	K. Y. Lim & P. Walsh
WD 5	Using JSD as a Description Language	P. Walsh
WD 6	Specification of Direct Manipulation Interfaces	P. Walsh
WD 7	Description of the Macdraw User Interface using Action Effect Rules	P. Walsh
WD 8	On Building JSD Models from Informal User Requirements Specifications	P. Walsh
WD 9	An Assessment of the Macdraw Application Package	K. Y. Lim
WD 10	Applying JSD to the Specification of Human Factors Design Issues	P. Walsh
WD 11	An Anatomy of Software Methodology	K. Y. Lim
WD 12	On Terminology and Design Perspectives	K. Y. Lim
WD 13	Task Analysis Methods in HCI Design	P. Walsh
WD 14	Some Tentative Initial Thoughts Towards a Possible Framework for Modelling Systematic Methods of System Development	J. B. Long
WD 15	Task Strategies, Metaphors and the Integration of Interface Design into JSD	K. Y. Lim
WD 16	Proposed Plan of Action for Network Manager Case-Study	P. Walsh
WD 17	Essential Considerations for Interfacing Task Analysis and Task Modelling with JSD Modelling	K. Y. Lim
WD 18	On Reasoning about the Project and DDN Case-Study Proposals	K. Y. Lim
WD 19	On Reasoning about User Interface	

	Design using Extended JSD in Conjunction with an Extant System Systems Analysis Approach	K. Y. Lim
WD 20	The Analysis of Task Objects Method (ATOM) : A Method of Representing User Activities for User Interface Design	P. Walsh
WD 21	Functional Specification for the Network Manager Workstation	P. Walsh & K. Y. Lim
WD 22	Rationalization of the Project Developments Based on the JSD* Models Described in WD 19 and Infotech Paper 1	K. Y. Lim
WD 23	Some Considerations on Notational Requirements for Task and Interface Information Capture : Towards the Conception of a JSD* Notation	K. Y. Lim
WD 24	Reply To D&L's Human Computer Interaction Engineering (HCIE) Conception -- An Explication of the Corresponding Definitions of Human Factors (HF) Terms adopted by the RARDE Project and KYL's PhD Work	K. Y. Lim
WD 25	Explication of KYL's Conception of the Research Activities Required for the Derivation and Specification of the JSD* Method	K. Y. Lim
WD 26	A Perspective on HCI Model Classes and their Life-Cycle w.r.t. the System Design Process : Providing a Rational Basis for the Current Conception of JSD*	K. Y. Lim
WD 27	Case-Study Illustrations of a Conception of Structured Interface Design Within JSD*	K. Y. Lim
WD 28	Towards a JSD* Method -- A Review of the Research Scope, Requirements and Constraints; and the Proceduralisation of the JSD* User Interface Design Process	K. Y. Lim
WD 29	A Preliminary Conception of the Stage-Wise Process of User Interface Design within JSD* : A Case-Study Instantiation using the University College London	

	Recreation Facility Booking System (UCL RFBS)	K. Y. Lim
WD 30	Towards a Task Analysis Method for Interface Design within JSD* : The Usefulness of Generification Procedures	S. Coles & K. Y. Lim
WD 31	Towards a Task Analysis Method for JSD* : An Analytical and Procedural Review of Task Analysis for Knowledge Descriptions and Knowledge Analysis of Tasks	N. Silcock & K. Y. Lim
WD 32	Towards a Task Analysis Method for JSD* : An Analytical and Procedural Review of 'Hierarchical Task Analysis'; 'Hierarchical Planning' and 'GOMS'	N. Silcock
WD 33	An Overview of the First Pass Version of the JSD*(HF) Method	K. Y. Lim
WD 34	A First Pass Version of the Proceduralised JSD*(HF) Method	N. Silcock & K. Y. Lim
WD 35	An Appraisal of the Requirements of the Digital Data Network Network Management Case-Study Test -- Perspectives Corresponding to Client Simulation and Research Constraints	K. Y. Lim
WD 36	Event Tables for the DDN Case-Study	N. Silcock & K. Y. Lim
WD 37A	JSD*(HF) Deliverables Corresponding to the First Conjunction with the JSD*(SE) Design Stream -- Outputs of the CTM and SUN Stages for the Trouble-Shooting Module of the DDN NMS Case-Study	K. Y. Lim
WD 37B	JSD*(HF) Deliverables Corresponding to the First Conjunction with the JSD*(SE) Design Stream -- Outputs of the CTM and SUN Stages for the Security Module of the DDN NMS Case-Study	N. Silcock
WD 38A	The DDN NMS Case-Study -- Documenting Outputs of the ESSA and GTM Stages for the Trouble-Shooting Module and Associated Refinements Suggested for the JSD*(HF) Method	K. Y. Lim

WD 38B	The DDN NMS Case-Study -- Documenting Outputs of the ESSA and GTM Stages for the Security Module and Associated Refinements Suggested for the JSD*(HF) Method	N. Silcock
WD 39A	The DDN NMS Case-Study -- Documenting Outputs of the CTM and Post-CTM Stages for the Trouble-Shooting Module and Associated Refinements Suggested for the JSD*(HF) Method	K. Y. Lim
WD 39B	The DDN NMS Case-Study -- Documenting Outputs of the CTM and Post-CTM Stages for the Security Module and Associated Refinements Suggested for the JSD*(HF) Method	N. Silcock & K. Y. Lim
WD 40A	JSD*(HF) Specifications of the User Interface for the DDN NMS Case-Study (Trouble-Shooting Module)	K. Y. Lim

Table B-1 : Authorship Summary for RARDE Project Working Documents

Author of Document	Total Number of Reports	Working Document Number
Lim	22	WD 1*, 9, 11, 12, 15*, 17, 18*, 19*, 22*, 23, 24, 25*, 26*, 27*, 28*, 29*, 33*, 35*, 37A*, 38A*, 39A*, 40A*.
Lim & Walsh	1	WD 4.
Coles & Lim	1	WD 30.
Silcock & Lim	4	WD 31, 34*, 36, 39B.
Walsh & Lim	3	WD 2*, 3, 21.
Long	1	WD 14
Silcock	3	WD 32, 37B, 38B.
Walsh	8	WD 5, 6, 7, 8, 10, 13, 16, 20.

Total Number of Working Documents = 43

(Major project deliverables are marked with an asterisk)

(C) Cumulative List of PhD Working Documents

PhD 1 Draft Proposal of a PhD Project entitled Interface Design Derivation via an Extant System Systems Analysis Approach

PhD 2 Relationship between the Areas of Concerns of the PhD and RARDE Projects -- the Overlap and Extensions

PhD 3 Hierarchical Description of the Structure of Overall Conduct of the PhD Research and an Overview of the Proposed Case-Studies and Experimental Investigations

PhD 4 Preliminary Impressions on Sutcliffe's Report on Extending JSD to include Human Factors

PhD 5 The Design and Psychological Rationale Underlying the JSD*(HF) Method