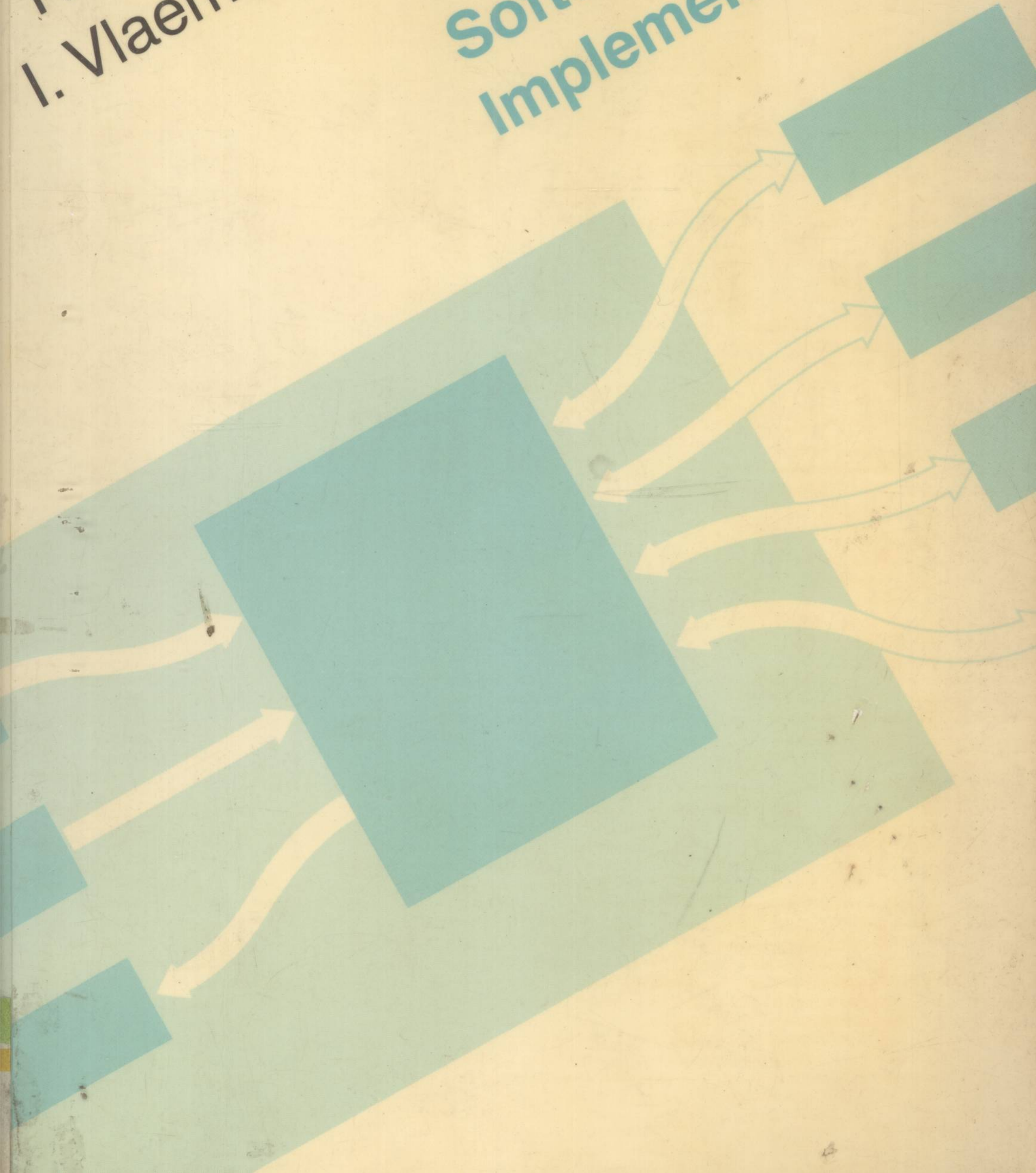
Blackwell Scientific Publications

Computer Science Texts

R.B. Coats
I. Vlaeminke

Man-Computer Interfaces: An Introduction to Software Design and Implementation



COMPUTER SCIENCE TEXTS

Man–Computer Interfaces

R. B. COATS
and
I. VLAEMINKE
Both Principal Lecturers
in Computing
Leicester Polytechnic

BLACKWELL SCIENTIFIC PUBLICATIONS

OXFORD LONDON EDINBURGH

BOSTON PALO ALTO MELBOURNE

© 1987 by
Blackwell Scientific Publications
Editorial offices:
Osney Mead, Oxford OX2 0EL
(*Orders*: Tel. 0865 240201)
8 John Street, London WC1N 2ES
23 Ainslie Place, Edinburgh EH3 6AJ
52 Beacon Street, Boston
Massachusetts 02108, USA
667 Lytton Avenue, Palo Alto
California 94301, USA
107 Barry Street, Carlton
Victoria 3053, Australia

First published 1987

Set by V & M Graphics Ltd,
Aylesbury, Bucks
Printed and bound in Great Britain by
Mackays of Chatham, Kent

DISTRIBUTORS

USA and Canada
Blackwell Scientific Publications Inc
PO Box 50009, Palo Alto
California 94303
(*Orders*: Tel. (415) 965-4081)

Australia
Blackwell Scientific Publications
(Australia) Pty Ltd
107 Barry Street
Carlton, Victoria 3053
(*Orders*: Tel. (03) 347 0300)

British Library
Cataloguing in Publication Data
Coats, R. B.
Man–computer interfaces: an introduction to software design and implementation.
1. Computer interfaces
I. Title II. Vlaeminke, I.
004.6 TK7887.5

ISBN 0–632–01542–X

Library of Congress
Coats, Robert B.
Man–computer interfaces: an introduction to software design and implementation/R. B. Coats & I. Vlaeminke.
Includes index.
ISBN 0–632–01542–X
1. Computer software—Development. 2. Electronic data processing—Psychological aspects. 3. System design.
I. Vlaeminke, I.
II. Title
QA76.76.D47C63 1987
005—dc19 87–16110

Preface

The last ten years have seen an increasing recognition of the important role played by the man–computer interface in the success of a computer system. Interest in this aspect of systems development has grown not only amongst computer scientists but also amongst ergonomists, psychologists, sociologists and graphic designers, reflecting its multi-disciplinary nature. Indeed, it has frequently been suggested that for many applications, successful development requires a design team whose members represent these various disciplines.

This text is aimed at undergraduates following a course in Computer Science, the future providers of the software expertise in such a multi-disciplinary team. It focuses on the facilities which can be provided by software in the interface and on how they can be implemented, and proposes a strategy for this aspect of systems design. It provides an introduction to the basic facilities and concepts of text-based interfaces in the conviction that competence in these basics is a prerequisite to an understanding of more advanced interfaces and to an appreciation of the possibilities for further advances. In the authors' opinion, graphical interaction deserves a separate treatment and is not covered, other than in the obvious overlap in window-based systems.

The major theme underlying the book is that the interface software can be identified as a distinct element which can be separated from the task processing in any system. Furthermore, the interface can be subdivided into a number of layers, whose functions can be represented by generalised abstractions. The concept of abstraction is introduced by deriving abstractions for basic input and output processes, and illustrating their practical implementation. A common abstraction is then developed for the traditional dialogue structures (Question and Answer, menus, forms and commands) to demonstrate that these represent simple variants of the same basic structure. This interpretation is used to explain the features of each structure and to justify accepted guidelines on their use.

The dialogue abstractions are extended to incorporate techniques which ameliorate the impact of system response on users of the system and which permit a limited form of adaptation, within a system, to the requirements and preferences of different users. The input and output abstractions are extended to systems which support multiple, overlapping windows and a direct manipulation style of interface.

The identification of these abstractions is fundamental to the proposed design strategy since, by encouraging the modularisation of the interface and the development of module libraries, it improves consistency and portability and permits a prototyping approach to development. Transition network and production system representations are described to illustrate that dialogues based on the abstractions can be generated automatically.

Although the emphasis is on software techniques and a design strategy for their utilisation, accepted guidelines for other aspects of interface design are explored, and an attempt is made to explain the rationales which underpin them. The level of detail is based on the criterion that the 'software member' of the design team must be capable of appreciating inputs from other disciplines in deciding the data values to be supplied to the abstractions in different situations. The final chapter provides a brief introduction to more advanced interfaces which extend the range of input/output channels or the adaptation to different user characteristics.

Practical illustration of how the abstractions can be implemented requires a choice of programming language and of workstation facilities. The language chosen is PASCAL, not because it represents the optimal choice for implementation, but because it is commonly used to teach programming and because its syntax is easily readable by someone familiar with other procedural languages; the ideas illustrated in PASCAL are easily transferred to other langauges such as Modula-2, C or APL. The workstation assumed in most examples is one with facilities equivalent to a PC-compatible equipped with a mechanical mouse. Again this choice is dictated by popularity, rather than innate virtue, and because these facilities can be considered a base standard for a typical workstation; examples are given of the facilities offered both by dumb terminals and by more sophisticated workstations.

The practical exercises which accompany a course on programming typically require some design of input and output, but programming language texts necessarily concentrate on the mechanics of the input and output functions provided within the particular programming language. This book is designed to complement a programming course by discussing the enhancements to these raw input and output functions, both in structure and in facilities, which are necessary for a successful interface, and by giving practical illustrations of how they can be accomplished. Most chapters are followed by sets of discussion and programming exercises to provide practical exposure to the concepts. A design exercise for a simple transaction processing system runs through the early chapters. The Appendices contain two other case studies and listings of all machine-independent code referenced in the text.

The book is intended as an introductory text; it is not a summary of the latest research findings. The bibliographies which accompany each chapter are not comprehensive but are intended to provide a next step in following up the concepts introduced. It is the authors' conviction that an appreciation of the published guidelines for interface design is best achieved by experimenting with different types of interface; for example, the reader should experience for himself the confusion which results when a screen is filled with a multiplicity of different highlights. For this reason, the text includes only a limited number of illustrations drawn from commercially available systems; these are used to illustrate specific techniques rather than as exemplars of interface design. Interaction with a system is a dynamic process whose characteristics are not easily assessed from a static snapshot; at the end of various chapters, the reader is specifically encouraged to try out examples of typical packages to experience these dynamics.

The authors gratefully acknowledge the assistance of Apollo Computer Inc., CASE plc., City Business Systems and Digital Research Ltd. for their permissions to copy illustrations. Thanks are also due to a number of colleagues, and in particular to Ian Marshall, for their suggestions on revisions to the text.

Acknowledgements

Apollo and Domain are registered trademarks of Apollo Computer Inc.
CP/M, GEM and Desktop are registered trademarks of Digital Research Inc.
IBM is a registered trademark of International Business Machines Corp.
Lotus and 1-2-3 are registered trademarks of Lotus Development Corp.
MS-DOS is a registered trademark of Microsoft Corp.
UNIX is a registered trademark of Bell Laboratories.
WordStar is a registered trademark of MicroPro International Corp.

Contents

Chapter 1

The significance of the man–computer interface

1.1 What is the Interface?

Every computer system can be measured against two criteria:

Correctness and **Convenience**

With the traditional concept of a computer system as illustrated in Fig. 1.1, correctness means that if suitable input values are supplied to the process it will produce the desired output values.

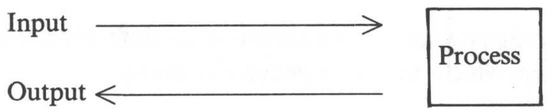


Fig. 1.1.

Systems staff have traditionally emphasised this computational correctness; much effort has been applied to ways both of producing software and of testing it in an attempt to achieve correctness. Whilst few, if any, of these would claim to have produced commercial software which operates 100% correctly, the majority of systems perform reasonably well against this criterion, and most are at least consistent in the results they produce.

Much less effort has been devoted to the concept of convenience. Although any computer system includes at least one human user within its boundary, systems have been viewed as undertaking a task — such as maintaining a sales ledger — rather than as a tool used by a human to assist in achieving that task. It is the sales clerks who maintain the company's sales ledger not the computer system! The emphasis on the correctness of the values has tended to be at the expense of considering exactly how these values should be input or output; function has been the significant factor rather than form.

The input and output elements are too often viewed by software producers as a series of rather tedious read and write operations which must be gone through before one gets to 'the interesting bit of the program'; have you ever heard a programmer claim that they 'produced a really great write statement today'? However, the results of these read and write operations, and the terminal devices on which they occur, are often all the human user sees of the computer system.

The user of a computer-based system has a right to expect not only that the system produces correct values but also that it is easy to use. This implies that the human does not have to alter his natural work pattern significantly in order to use the computer system. In most cases, the actual processing of the input values to produce the output values has no impact on this since the user does not experience it directly and has no need, or desire, to know how it is done. The actual mechanics by which the steering of a car is transferred from steering wheel to the road wheels is of little consequence to the motorist; on the other hand, the shape and position of the steering wheel has an enormous impact on him. Similarly, the nature and positioning of the workstation, the format in which input is requested, and the layout of messages produced by the system, have an enormous impact on the human user of the computer system.

The *man–computer interface* encompasses all those aspects of a computer-based system which the user experiences directly.

1.2 What Factors Affect Convenience?

For his usage to be 'convenient', the user must feel 'comfortable' when interacting with the system. Thus, the factors which affect convenience are those which influence this feeling of comfort. They can be divided into the three broad classifications of Fig. 1.2.

Classification	*Affected by*	*Influences*
social factors	organisational climate	emotional comfort
physical ergonomics	hardware	physical comfort
psychological ergonomics	software design	cognitive comfort

Fig. 1.2. Factors influencing a user's comfort.

The general climate within the organisation — such aspects as management style and job security — and the way in which a prospective system is introduced, can build preconceptions long before the actual system is encountered. The functions which are allotted to the user by the system, and the way in which these human functions must be carried out, can disrupt traditional social groupings in the workplace, isolate the user from normal social interaction or disturb relationships with superiors. These *social factors* will tend to reinforce or to allay a user's fears about a system. A great deal has been written about this aspect of system development and about the involvement of users in the development process. We will be concerned only with how the use of a particular software design strategy can facilitate this involvement.

Assuming that a user approaches a system with no negative preconceptions, the *ergonomics* or usability of the actual system can significantly improve or worsen his attitude towards it. The main aspects of ergonomics are shown in Figure 1.3.

— design and arrangement of equipment
— design of the dialogue
— availability and reliability of the system
— responsiveness of the system

Fig. 1.3. The ergonomic aspects.

The way in which the equipment, both computer hardware and any ancillary equipment, is designed and arranged into a workstation will affect the *physical comfort* of the user when using the system. Can the user read the characters on the screen easily, or has it been positioned so that the sun causes a glare which makes the user peer at the screen through half-closed eyes? Can the keyboard be positioned so that the user can reach the keys and any other items required without having to stretch unnaturally? For example, in some supermarkets the checkout tills are positioned in such a way that the cashiers have to twist and reach for each item being purchased — resulting in a lot of tired cashiers at the end of a day! Has the system designer given as much thought to the seating and worktop as to the terminal which rests upon it, or did he just use any old table and chair which happened to be spare? The design of equipment is the province of ergonomics; with most computer systems, the systems designer selects off-the-shelf equipment rather than designing it from scratch.

The second aspect of the interface might be termed *psychological ergonomics*; just as physical ergonomics is concerned with matching the system to human physical processes, psychological ergonomics is concerned with matching it to human cognitive processes. To illustrate the difference, consider the process of reading a message from a screen. A user who cannot physically discern the characters because of glare or poor contrast will suffer physical discomfort. The user who manages to read the characters but who cannot comprehend the message because it is phrased in computer jargon, or because it is laid out in some eccentric fashion, will suffer psychological discomfort. There is little point in sparing a user the discomfort of back ache or eye strain if the system makes his brain ache instead! This aspect, called the *dialogue*, is an area of systems design which software producers can very easily influence for good or ill, and its design is the major concern of this book.

Two aspects closely related to psychological ergonomics, but sufficiently important to warrant individual treatment, are the *availability* and the *responsiveness* of the system.

Can the user gain access whenever and wherever he needs to? The designer should ensure that the times when the system can be expected to be available match the hours when the user will require it. Furthermore, the reliability of the system must be such that the user can reasonably expect it to be available when it is supposed to be. It is not only the total amount of time lost due to faults which is significant but also the number of faults; a series of losses in a network link, each lasting no more than a few seconds, can be much more frustrating than a single failure lasting an hour. The number of workstations must be adequate to support the number of prospective users. Unless usage is very casual or work patterns are not affected if a workstation is not immediately available, this generally implies individual workstations located within the user's normal work area. There are a surprising number of systems which provide users with an instantaneous answer to a query after they have walked up two flights of stairs and queued for ten minutes for a workstation to become free. Economising on the number of workstations is a false economy.

Almost as frustrating as being unable to access the system at all, is being expected to wait 20 seconds or longer for the system to respond to the last input. Even worse is the case where some days it takes two seconds to respond and other days it takes 20 seconds; variable response is a marvellous way to keep the user guessing whether the system has crashed or not! The provision of acceptable response is one of the more technically demanding and costly aspects of interactive system development, and is discussed in Chapter 8.

1.3 Why is Convenience Significant?

In the 1960s and early 1970s the user's convenience was largely ignored by systems designers. The large mainframe was a precious item to be cosseted with air conditioning and an army of operators; humans fitted in with the machine rather than vice versa.

With the demise of batch processing came an increasing realisation that convenience was a major factor in the success or failure of a system; this awareness was heightened as systems development moved away from the clerical bread-and-butter data processing applications into the area of decision-support for management. For the system to be effective, it is not sufficient for the hardware and software to produce the correct output values for given input values — the human's performance is critical to success.

Humans have emotional, physiological and psychological needs which must be met in any activity if that activity is to be performed effectively. A user who is confused, frustrated or stressed physically or psychologically *cannot* perform well. The component of a system which causes the presence or absence of stress is the man–computer interface — that is what the user experiences when interacting with the system. Some cynics might observe that this awareness has yet to dawn on the designers of many microcomputer operating systems!

The human body is a mechanism which has limitations and tolerances within which it must work. Our eyes require images to be within a particular size range, of a certain level of brightness, to contrast sufficiently against their background, and to be located a suitable distance away if they are to be viewed in comfort. Some colours are perceived more easily than others, some colour juxtapositions aid discrimination whilst others are confusing. We can move our limbs only over certain ranges, our reach is limited, and our hands have limits on their dexterity. If we are to maintain a particular position for any length of time our bodies need adequate support, and so on.

These physical limitations are commonly recognised. However, the limitations of our brains are less well understood and are more often overlooked. Whilst we have an extremely capacious *long term memory*, we have a very limited *short term memory*. Short term memory is often considered as a series of input and output buffers in which intermediate data can be stored during any activity. Like the buffers in a computer system, this memory has a very limited capacity and can be easily overloaded. Long term memory seems to have an unlimited capacity and humans can retrieve information from it very rapidly. If we undertake an activity regularly, we can easily 'remember' it without overloading the short term memory; if we do it often enough, it becomes almost subconscious. However, if we carry out an activity irregularly, our short term memory is fully occupied throughout the activity.

Even if the basic activity itself is familiar, we can overload the short term memory with a particular instance of that activity. Consider the case of a clerk looking up the details of an invoice. These details are displayed across two screens because there is too much detail to fit onto one. The clerk may have carried out the task many times, and may well be able to tell you without difficulty which fields are on which screen, since he will have seen the same field captions many times. Remembering the values for a particular invoice from one screen to the next will be much more difficult, if not impossible.

Humans bring to every activity a set of expectations of how that activity should proceed. These expectations — a mental *model* of the activity — are based on their previous experience. Long term memory is often viewed as a store of patterns representing different models against which humans seek to match mental stimuli; rather than storing low level details, humans reconstruct this detail information from higher level patterns. Most people expect to read a screen from left to right and from top to bottom, just as they read a printed page. They look for order and structure in a display, and seek clues as to the relative importance of different items.

Humans are remarkably adaptable. They can contort themselves into weird positions and operate under the most unfavourable conditions — just look at programmers at their workstations! They can supplement their short term memory with pieces of paper. They can adapt the way in which they undertake a task. They can acquire new models which run contrary to their previous expectations and impose order in a display where none existed. But this adjustment causes stress which may exhibit itself in confusion, frustration or physical aches and pains.

The results of this stress can take various forms. Where usage is discretionary the user may simply opt not to use the system. For example, take the situation where a manager could use a financial planning system or he could do the exercise manually using a calculator. If his perceptions of the system are unfavourable because of poor availability, response, physical discomfort, or whatever, he might choose the manual solution. Where the user has no choice, for example a clerk operating a sales ledger or order processing system, it might result in illness or persistent absence, or in an unacceptable level of errors. These errors are not deliberate attempts to sabotage the system — it is possible to arrange the inputs and outputs of a computer system so that it is difficult to get them right!

Many systems in use today exhibit problems caused by failure of the interface to meet the users needs in these areas. They were not deliberately designed to do so. Thus, one must examine why so many have failed.