

CAMBRIDGE SERIES ON HUMAN-COMPUTER INTERACTION

FORMAL METHODS IN HUMAN- COMPUTER INTERACTION

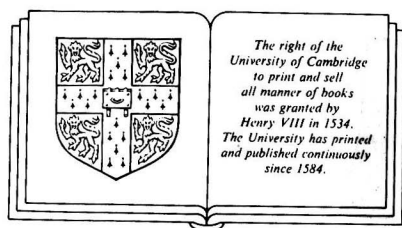
EDITED BY
M. HARRISON & H. THIMBLEBY

7811 Δ
H321

9063569

Formal Methods in Human-Computer Interaction

Edited by Michael Harrison
University of York
and
Harold Thimbleby
University of Stirling



E9063569

CAMBRIDGE UNIVERSITY PRESS

Cambridge
New York Port Chester
Melbourne Sydney

Published by the Press Syndicate of the University of Cambridge
The Pitt Building, Trumpington Street, Cambridge CB2 1RP
40 West 20th Street, New York, NY 10011, USA
10, Stamford Road, Oakleigh, Melbourne 3166, Australia

© Cambridge University Press 1990

First published 1990

Printed in Great Britain at the University Press, Cambridge

Library of Congress Cataloguing in Publication data available

British Library Cataloguing in Publication data available

ISBN 0-521-37202-X

CAMBRIDGE SERIES ON HUMAN-COMPUTER INTERACTION 2

**Formal Methods in
Human-Computer Interaction**

Cambridge Series on Human-Computer Interaction

Managing Editor: Professor J. Long,
Ergonomics Unit, University College, London.

Editorial Board

Dr P. Barnard, Medical Research Council,
Applied Psychology Unit, Cambridge, UK
Professor H. Thimbleby, Department of Computing Science,
University of Stirling, UK
Professor T. Winograd, Department of Computer Science,
Stanford University, USA
Professor W. Buxton, Rank Xerox Ltd, Cambridge EuroPARC, UK
Dr T. Landauer, Bellcore, Morristow, New Jersey, USA
Professor J. Lansdown, CASCAAD, Middlesex Polytechnic, UK
Professor T. W. Malone, MIT, Cambridge, Massachusetts, USA
Dr J. Grudin, MCC, Austin, Texas, USA

Titles in the Series

1. J. Long and A. Whitefield *Cognitive Ergonomics and Human-Computer Interaction*
2. M. Harrison and H. Thimbleby *Formal Methods in Human-Computer Interaction*

PREFACE

This is the first book specifically to relate modern, formal, ideas in Software Engineering to Human Computer Interaction. The book is intended to be read by software engineers, HCI researchers, and post-graduate students working in or with HCI and Software Engineering.

By collecting and representing the state of the art in relevant HCI research, this book addresses the question of how software systems can be designed and built that incorporate a full consideration of the user. Formal design methods should capture the perspective of the user within a software engineering framework.

Our aim is to contribute to both HCI and formal methods by applying one to the other, in particular, by showing how formal methods may be used to model and implement prototypes of interactive systems. The material, then, is of advantage to people working in conventional HCI—we expose them to the power and relevance of formal methods—and conversely, to people working in formal methods—we expose them to the applications and potential in HCI.

Chapters 2 and 3 illustrate the gulf between software engineering and HCI. Subsequent chapters first show how formal modelling techniques may be used to describe interactive behaviour, and discuss how these models may be used to assist the design process (chapters 4, 5 and 6) and then discuss the relationship between models and implementations:

rapidly developed prototypes on the one hand; and system architectures on the other (chapters 7, 8 and 9).

A note on producing this book

This book was produced using $\text{\LaTeX}$, a system that enabled us to collate and edit the contributions and work at two distant sites in the UK, exchanging manuscripts and corresponding by email. $\text{\LaTeX}$ produces very good results when it works; for our purposes, it was better than alternatives—but it would have been *much* better for want of a formal model!

Acknowledgements

The work collected here represents an outgrowth of the activities of the Human Computer Interaction Group at York, both through research carried out there since 1983 and workshops, colloquia and conferences organised by the editors. We are particularly grateful to members of our research groups for providing stimulating working environments—particularly Chris Roast who helped with the diagrams and Chris Johnson who read the penultimate version.

MacDraw and MacWrite are registered trademarks of Claris Corporation. MacIntosh is a registered trademark of Apple Computer Inc. Multiplan is a registered trademark of Microsoft Corporation. UNIX is a registered trademark of AT & T Bell Laboratories.

CONTRIBUTORS

Heather Alexander

British Telecom PLC,
Exchange House,
229, George Street,
Glasgow, G1 1B2,
Scotland.

Gilbert Cockton

Department of Computer Science,
University of Glasgow,
17, Lilybank Gardens,
Glasgow, G12 8QQ,
Scotland.

Alan Dix

Human Computer Interaction Group,
Department of Computer Science,
University of York,
Heslington,
York, YO1 5DD,
England.

-
- Thomas Green* MRC Applied Psychology Unit,
15, Chaucer Road,
Cambridge, CB2 2EF,
England.
- Michael Harrison* Human Computer Interaction Group,
Department of Computer Science,
University of York,
Heslington,
York, YO1 5DD,
England.
- Jifeng He* Programming Research Group,
Oxford University Computing Laboratory,
8-11, Keble Road,
Oxford, OX1 3QD,
England.
- Colin Runciman* Department of Computer Science,
University of York,
Heslington,
York, YO1 5DD,
England.
- Franz Schiele* GMD-IPSI,
Cognitive User Interface Group,
Dolivostr. 15,
D-6100, Darmstadt,
West Germany.
- Bernard Sufrin* Programming Research Group,
Oxford University Computing Laboratory,
8-11, Keble Road,
Oxford, OX1 3QD,
England.
- Harold Thimbleby* Department of Computing Science,
University of Stirling,
Stirling, FK9 4LA,
Scotland.

Roger Took

Human Computer Interaction Group,
Department of Computer Science,
University of York,
Heslington,
York, YO1 5DD,
England.

CONTENTS

Preface	xiii
Contributors	xv
1 The role of formal methods in human-computer interaction	1
<i>Michael Harrison and Harold Thimbleby</i>	
1.1 Introduction	1
1.2 Some examples	2
1.3 The scope of formal methods	4
1.4 Book structure	5
1.5 The future	7
2 HCI formalisms and cognitive psychology: the case of Task-Action Grammar	9
<i>Franz Schiele and Thomas Green</i>	
2.1 Introduction	9
2.2 The notion of consistency	11
2.3 Task-Action Grammar to analyse consistency	14
2.3.1 An informal account	14

2.3.2	A formalisable account	16
2.3.3	TAG as a psychological theory	19
2.3.4	TAG as a predictive tool	20
2.4	Related research using rule-schemas	23
2.4.1	An executable Task-Action Grammar	23
2.4.2	Recognition of users' plans	24
2.4.3	Inclusion of system responses	25
2.4.4	Formalising the underlying semantics	25
2.5	Applying TAG to life-size examples	26
2.6	Using TAG to capture cross-applicational consistency	31
2.7	The psychological credibility of lifesize TAG	36
2.7.1	TAG representations are fragile	37
2.7.2	Restrictions of features	38
2.7.3	Alteration of task features	39
2.7.4	Unrepresented knowledge	40
2.8	Formalisms in HCI: some evaluative remarks	43
2.8.1	Revelation	43
2.8.2	Prediction	45
2.9	Appendix: Task-Action Grammars	47
2.9.1	How to read a TAG	47
2.9.2	MacWrite	48
2.9.3	Dictionary of simple tasks in MacWrite	48
2.9.4	Rule schemas in MacWrite: task rules	49
2.9.5	Rule schemas in MacWrite: subtask rules	50
2.9.6	Multiplan	50
2.9.7	Dictionary of simple tasks in Multiplan	51
2.9.8	MacDraw	55
2.9.9	Dictionary of simple tasks in MacDraw	55
2.9.10	Common rules ('MacGeneric')	59
2.9.11	Rule schemas in MacGeneric: task rules	59
2.9.12	Rule schemas in MacGeneric: subtask rules	60
2.9.13	Common rules in Multiplan and MacDraw	61
3	Putting design into practice: formal specification and the user interface	63
	<i>Roger Took</i>	
3.1	Introduction	63
3.1.1	Presenter	65

3.2	The design process	66
3.2.1	Software engineering and formal methods	66
3.2.2	Formal notations and abstraction	67
3.2.3	Notation and design	68
3.2.4	A reflection on design in Z	71
3.2.5	Differentiation	72
3.2.6	Structuring	80
3.2.7	The user interface	82
3.3	Constraints on design	83
3.3.1	Soft constraints	83
3.3.2	Firm constraints	85
3.3.3	Hard constraints	87
3.3.4	Environmental constraints	89
3.3.5	Limitations of formal specification	94
3.4	Conclusions	96
4	Non determinism as a paradigm for understanding the user interface	97
	<i>Alan Dix</i>	
4.1	Introduction	97
4.2	Unifying formal models with non determinism	98
4.2.1	The PIE model	99
4.2.2	Problems for temporal systems	100
4.2.3	Problem for windowed systems	101
4.3	Non deterministic PIEs	102
4.3.1	Use for temporal systems	105
4.3.2	Use for windowed systems	105
4.3.3	Non deterministic properties of PIEs	106
4.3.4	Summary—formal models and non determinism	107
4.4	Non deterministic computer systems?	107
4.4.1	The tension for the user	108
4.4.2	Levels of non determinism	108
4.4.3	Behavioural non determinism	109
4.5	Sources of non determinism	110
4.5.1	Non determinism due to timing	111
4.5.2	Non determinism due to sharing	112
4.5.3	Data uncertainty	113
4.5.4	Procedural uncertainty	113

4.5.5	Memory limitations	114
4.5.6	Conceptual capture	114
4.5.7	Discussion	115
4.6	Dealing with non determinism	115
4.6.1	Avoid it	116
4.6.2	Resolve it	117
4.6.3	Control it	118
4.6.4	Use it	120
4.6.5	Summary—informal analysis	121
4.7	Deliberate non determinism	121
4.7.1	Static and dynamic consistency	122
4.7.2	Intermittent update	123
4.7.3	Declarative interfaces	124
4.7.4	Non deterministic intermittent update	125
4.7.5	Is it a good idea?	125
4.8	Discussion	126
5	A state model of direct manipulation in interactive systems	129
	<i>Michael Harrison and Alan Dix</i>	
5.1	Introduction	129
5.2	Direct manipulation	130
5.3	Chapter plan	131
5.4	A formal framework for direct manipulation	131
5.4.1	Principles	131
5.4.2	The interaction model	132
5.5	The relationship between input and command	135
5.5.1	Temporal ordering	135
5.5.2	Contextual problems	136
5.5.3	Adding structure to the input model	138
5.6	Mapping state to display	138
5.6.1	Display resolution	141
5.6.2	Partiality	143
5.7	Localising properties of direct manipulation systems	145
5.7.1	Visibility	146
5.7.2	Local properties when commands are visible	147
5.7.3	Exception models	148
5.8	Conclusions	150

5.9	Acknowledgements	151
6	Specification, analysis and refinement of interactive processes	153
	<i>Bernard Sufrin and Jifeng He</i>	
6.1	Introduction	153
6.2	Processes	154
6.2.1	The simple model	155
6.2.2	Traces	158
6.2.3	The need for an improved model	158
6.2.4	The improved model	159
6.2.5	Failures	161
6.2.6	Sequential processes	162
6.2.7	Constructive specification of traces	163
6.3	Interactive processes	167
6.3.1	Experimenting with views and results	171
6.3.2	Equivalence of command sequences	174
6.3.3	Side-effects	175
6.3.4	Restartability	175
6.3.5	Relating views to results	176
6.3.6	Undoing	178
6.4	Interactive process refinement	179
6.4.1	A refinement ordering	179
6.4.2	Properties preserved by refinement	181
6.4.3	Verification	183
6.4.4	A strategy for refinement	185
6.5	Specifying processes in Z: an example	186
6.5.1	Text manipulation	186
6.5.2	Translating to a process	189
6.5.3	Display and mouse	191
6.5.4	Putting the components together	192
6.5.5	Analysis of the editor	195
6.5.6	Summary	197
6.6	Further work	197
6.7	Acknowledgements	198
6.8	Glossary	199

7	From abstract models to functional prototypes	201
	<i>Colin Runciman</i>	
7.1	Introduction	201
7.2	Functional programming	202
7.2.1	Recursively defined functions over lists	203
7.2.2	Higher order functions	204
7.2.3	Infinite lists and lazy evaluation	204
7.2.4	Strictness	205
7.2.5	Reasoning about programs	206
7.2.6	Transformation of programs	207
7.3	The <i>PiE</i> model	209
7.3.1	Displays and results	210
7.3.2	Uses of the model	211
7.4	<i>PiE</i> as a higher order function	211
7.4.1	<i>PiE</i> enrichments and composed partial applications	213
7.4.2	Specific <i>PiE</i> examples	215
7.5	Transformational refinement	217
7.5.1	Transforming the model alone	218
7.5.2	Model shifting and state-machine specialisation .	224
7.5.3	Application-specific transformation	230
7.6	Summary and conclusion	231
8	Designing abstractions for communication control	233
	<i>Gilbert Cockton</i>	
8.1	The need for specialised software tools.	233
8.2	Architecture and abstraction	235
8.3	Tooling the user interface	238
8.3.1	Communication control	239
8.4	Requirements for communication control	240
8.4.1	User requirements for communication control . .	240
8.4.2	Designer requirements for communication control	242
8.4.3	Satisfying requirements	244
8.5	A new communication control abstraction	247
8.5.1	Generative transition networks: fundamentals . .	247
8.5.2	A notation for GTNs	251
8.5.3	Example GTN specifications	254
8.5.4	Remarks on the examples	261

8.6	GTNs as communication control abstractions	262
8.7	The casting requirements	265
8.7.1	Choice of abstraction and notation	266
8.7.2	Evaluation of abstraction and notation	266
8.7.3	Iterate or terminate?	267
8.8	Summary	270
9	Structuring dialogues using CSP	273
	<i>Heather Alexander</i>	
9.1	Introduction	273
9.2	Specifying user interfaces	275
9.3	Introduction to CSP	276
9.4	Examples	278
9.4.1	Example: a menu-based system	278
9.4.2	Example: concurrent dialogues	281
9.5	Executing CSP specifications	283
9.6	A family of dialogue design tools	284
9.6.1	Dialogue outlines	284
9.6.2	Dialogue scenarios	288
9.7	Dialogue prototypes	292
9.8	Discussion and conclusions	294
	Bibliography	297
	Index	317