

Cambridge University Press

0521347920 - A Practical Handbook for Software Development - N. D. Birrell and M. A. Ould

Frontmatter

[More information](#)

A practical handbook for software development

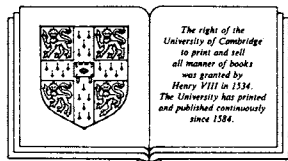
A practical handbook for software development

N. D. BIRRELL

Logica Pty Limited, Sydney, Australia

M. A. OULD

Logica UK Limited, London, UK



CAMBRIDGE UNIVERSITY PRESS

CAMBRIDGE

LONDON NEW YORK NEW ROCHELLE

MELBOURNE SYDNEY

Cambridge University Press

0521347920 - A Practical Handbook for Software Development - N. D. Birrell and M. A. Ould

Frontmatter

[More information](#)

Published by the Press Syndicate of the University of Cambridge
The Pitt Building, Trumpington Street, Cambridge CB2 1RP
32 East 57th Street, New York, NY 10022, USA
10 Stamford Road, Oakleigh, Melbourne 3166, Australia

© Cambridge University Press 1985

First published 1985

Reprinted 1986

Library of Congress catalogue card number: 84-9403

British Library cataloguing in publication data

Birrell, N.D.

A practical handbook for software development.

1. Electronic digital computers-programming

I. Title II. Ould, M. A.

001.64'25 QA76.6

ISBN 0 521 25462 0 Hardback

ISBN 0 521 34792 0 Paperback

Transferred to digital printing 2003

Contents

<i>Foreword</i>	ix	2.4.1 Introduction to the problem	20
<i>Preface</i>	x	2.4.2 Baselines	20
1 The choices facing the system developer	1	2.4.3 Issuing deliverable items	21
2 The software development process	3	2.4.4 The mechanisms of configuration control	22
2.1 Phases and stages	4	3 Development techniques	24
2.1.1 Delineating the phases – deliverables	5	3.1 Introduction	24
2.1.2 The phase profile	6	3.2 Cost estimation models	31
2.1.3 The Project Inception Phase	6	3.3 Structured English	32
2.1.4 The System Definition Phase	6	3.4 PSL/PSA	32
2.1.5 The System Design Phase	7	3.5 Structured Analysis – SA	33
2.1.6 The System Production Phase	7	3.6 Structured Analysis and Design Technique – SADT	34
2.1.7 The System Acceptance Phase	7	3.7 Controlled Requirements Expression – CORE	34
2.1.8 The Post-Acceptance Development Phase	8	3.8 Software Requirements Engineering Methodology – SREM	35
2.1.9 The Project Debriefing Phase	8	3.9 Finite State Machines – FSM	36
2.1.10 The Work Breakdown Structure	9	3.10 Petri Nets	37
2.1.11 Prototyping	10	3.11 Jackson System Development – JSD	37
2.2 Quantitative software development models	12	3.12 RSRE Software Development System – SDS/RSRE	38
2.2.1 Introduction	12	3.13 Structured Design – SD	39
2.2.2 Approaches to estimating	13	3.14 Jackson Structured Programming – JSP	39
2.2.3 Algorithmic estimation methods	14	3.15 System Architect’s Apprentice – SARA	40
2.2.4 Allocating effort to phases	15	3.16 MASCOT	40
2.2.5 Deriving the development schedule	16	3.17 Formal Development Methodology – FDM	41
2.2.6 Is reliability quantifiable?	17	3.18 Hierarchical Development Methodology – HDM	42
2.2.7 Using historical data	18	3.19 Higher Order Software – HOS	42
2.3 Indicators	18		
2.3.1 Why indicators?	18		
2.3.2 Monitoring indicators	18		
2.4 Configuration management	20		

Contents

vi

3.20	Structured programming	43	5.4.7	Finite State Machines – FSM	82
3.21	Control of data	44	5.4.8	Petri Nets	87
3.22	Coding techniques	44	5.4.9	Jackson System Development – JSD	91
3.23	Unit testing techniques	45	5.4.10	Software Development System – SDS/RSRE	95
3.24	System integration	45	5.4.11	Structured Walkthroughs	98
3.25	Structured Walkthroughs	45	5.5	Indicators to be monitored during System Definition	100
3.26	Fagan inspections	46	5.5.1	Source code size estimate	100
4	Project Inception	47	5.5.2	System size indicator	100
4.1	The purpose of the Inception Phase	47	5.5.3	Requirement Specification quality indicator	101
4.2	The starting point of the Inception Phase	47	5.6	Checklists for the System Definition Phase	101
4.3	The finishing point of the Inception Phase	48	5.6.1	Quality criteria for the Functional Specification	101
4.4	Techniques available during Project Inception	49	5.6.2	Contents of the Functional Specification	103
4.5	Looking forward	49	5.6.3	Conduct of the Acceptance Test	104
4.5.1	Tender evaluation	49	5.6.4	System Definition and maintainability	104
4.5.2	Tender preparation	50	5.6.5	Contents of the Quality Management Plan	105
4.6	Checklist for the Project Inception Phase	50	5.7	Memorabilia for the System Definition Phase	106
4.7	Memorabilia during Project Inception	51	6	System Design	110
4.8	A sample case-history – VISTA	51	6.1	The purpose of the System Design Phase	110
4.8.1	Introduction	51	6.1.1	Introduction	110
4.8.2	Image processing background	52	6.1.2	Ease of implementation	110
4.8.3	VISTA functionality and hardware configuration	54	6.1.3	Functional completeness	111
5	System Definition	56	6.1.4	Compliance with constraints	111
5.1	The aims of the System Definition Phase	56	6.1.5	Ease of maintenance	112
5.2	The starting point of System Definition	56	6.2	The starting point of System Design	112
5.3	The finishing point of System Definition	56	6.3	The finishing point of System Design	113
5.3.1	The Functional Specification	56	6.3.1	Structure	113
5.3.2	The Project Plan	57	6.3.2	Detail	113
5.3.3	The Quality Management Plan	57	6.3.3	Validity	114
5.3.4	The Acceptance Test Specification	58	6.3.4	The deliverable item	114
5.4	Techniques available during System Definition	58	6.4	Techniques of use during System Design	114
5.4.1	Structured English	60	6.4.1	Structured Design – SD	115
5.4.2	PSL/PSA	62	6.4.2	Jackson Structured Programming – JSP	120
5.4.3	Structured Analysis – SA	64	6.4.3	Jackson System Development – JSD	127
5.4.4	Structured Analysis and Design Technique – SADT	68	6.4.4	Finite State Machines – FSM	130
5.4.5	Controlled Requirements Expression – CORE	71			
5.4.6	Software Requirements Engineering Methodology – SREM	77			

Contents

vii

6.4.5	Petri Nets	131	7.2	The starting point of System Production	169
6.4.6	System Architect's Apprentice – SARA	133	7.3	The finishing point of System Production	170
6.4.7	MASCOT	141	7.4	Techniques of use during System Production	170
6.4.8	Formal Development Methodology – FDM	145	7.4.1	Jackson Structured Programming – JSP	171
6.4.9	Hierarchical Development Methodology – HDM	148	7.4.2	Jackson System Development – JSD	172
6.4.10	Higher Order Software – HOS	152	7.4.3	Finite State Machines – FSM	175
6.4.11	Structured Walkthroughs	157	7.4.4	System Architect's Apprentice – SARA	175
6.4.12	Fagan inspections	158	7.4.5	MASCOT	176
6.5	Indicators to be monitored during System Design	159	7.4.6	Hierarchical Development Methodology – HDM	179
6.5.1	What we are looking for	159	7.4.7	Structured programming	180
6.5.2	An interconnectivity metric	159	7.4.8	Control of data	184
6.5.3	Myers' module coupling metric	160	7.4.9	Programming languages	187
6.5.4	Myers' module cohesion metric	160	7.4.10	Macroprocessors	189
6.5.5	Call graph metrics	161	7.4.11	Program libraries and reusability	190
6.5.6	Error detection metric	161	7.4.12	Walkthroughs and inspections	191
6.5.7	Error correction time metric	162	7.4.13	Static analysis	192
6.5.8	Documentation size metric	162	7.4.14	Symbolic execution	193
6.5.9	Source code size estimate	162	7.4.15	Assertion checking	195
6.5.10	System size indicator	162	7.4.16	Test data selection	196
6.6	Checklists for the System Design Phase	163	7.4.17	Test coverage analysis and software metrication	198
6.6.1	Topics to be covered in the design process	163	7.4.18	Development environments	199
6.6.2	General properties of the design documentation	164	7.4.19	Techniques for system integration	203
6.6.3	System Design and maintainability	164	7.4.20	System developer psychology	207
6.6.4	Potential design error types	165	7.5	Indicators to be monitored during System Production	209
6.7	Memorabilia for System Design	165	7.5.1	Detailed design indicators	210
7	System Production	167	7.5.2	Coding indicators	210
7.1	The aims of the System Production Phase	167	7.5.3	Testing indicators	213
7.1.1	The background to System Production	167	7.6	Checklists for the System Production Phase	214
7.1.2	Elaborating the overall design	167	7.6.1	Checklist for detailed design	215
7.1.3	Transforming correct designs into correct code	168	7.6.2	Checklist for coding	215
7.1.4	Checking the correctness of the transformation into code	168	7.6.3	Checklist for unit testing	215
7.1.5	Putting the system together	169	7.6.4	Checklist for User Documentation	216
7.1.6	Ancillary stages during System Production	169	7.6.5	Checklist for System Conversion Schedule	216
			7.6.6	Checklist for the Training Schedule	217

Contents

viii

7.6.7 Checklists for the Acceptance Test Description	217	8.5.1 Preparing for change	223
7.6.8 Potential coding error types	219	8.5.2 Monitoring and assessing change	224
7.7 Memorabilia for the Production Phase	219	8.5.3 The specification, design and implementation of changes	224
8 System Acceptance and Post-Acceptance Development	222	9 Project Debriefing	226
8.1 The aims of System Acceptance	222	9.1 The purpose of Project Debriefing	226
8.2 The starting point of System Acceptance	222	9.2 The framework for Project Debriefing	226
8.3 The finishing point of System Acceptance	223	9.3 Contents of a Debriefing Report	227
8.4 Carrying out the Acceptance Test	223	9.4 A checklist for the Project Debriefing Report	228
8.5 Managing Post-Acceptance Development	223	9.5 Sample Project Debriefing Report	228
		<i>Other bibliographies</i>	233
		<i>References</i>	235
		<i>Index</i>	257

Foreword

The cost of software development is now the dominating cost in computer systems. As hardware costs decline, improved efficiency of software production becomes the key target if further reductions in computing costs are to be achieved. Accordingly the Alvey Committee identified software engineering as one of the four underlying technologies for the UK national programme of cooperative research in information technology.

Coming as it does at the start of the Alvey programme, I welcome this book by Martyn Ould

and Nick Birrell, for it presents a picture of software engineering techniques that are already in existence and can therefore be exploited now. If the techniques set out in the book are widely applied we can look forward to a very significant improvement in the efficiency of software production.

*Brian Oakley, Director, Alvey Programme
London, Autumn 1983*

Preface

With the accelerating pace of work being done in the field of software engineering, it is becoming increasingly difficult to see the range of available techniques against a common background. Individual techniques are accessible throughout the literature but generally each is described in isolation. Comparison is difficult unless one has a strong idea of what one is looking for.

Our purpose in writing this book has therefore been to bring together, within a single framework, a variety of methods from all stages of software development and, in this way, to assist software engineers in particular in finding and evaluating the techniques appropriate to their own working environments. This should be regarded principally therefore as a book of signposts, leading the reader out into the field with a map, a compass and some reference points. By combining pointers to a selection of modern development practices with practical hints and checklists, all within one framework, we hope to fill the need for a vade-mecum to the software developer.

Managers with responsibility for software production will also, we hope, find much to stimulate their ideas about what modern practices have to offer and how their departments' efficiency and effectiveness could benefit. Students of computing science will be able to complement their more theoretical work by seeing the software development process as a day-to-day activity, a process that has managerial and sociological as much as purely technical aspects. As software developers we need to be aware of the work that has been done

already in the software engineering field and that is ready for us to exploit.

Much of the flavour and content of this book naturally derives from our work for Logica and Logica's own philosophy and practice, and we would like to acknowledge both this contribution and the encouragement and facilities that Logica has given.

We would very much like to thank the following people for their advice and help:

John Cameron (Michael Jackson Systems Limited)
Deborah Cooper and John Scheid (System Development Corporation)
Tony Curry (DMW Group Europe)
Dave Forsyth (British Aerospace)
R. P. Loshbough (TRW)
Pieter Mimno (Higher Order Software Inc.)
Peter Neumann (SRI International)
Martyn Thomas (Praxis Systems Limited)
Tony Ward (British Aerospace)
Robert Worden (Logica Limited)

Suzanne Wallace and her word-processing team (particularly Cheryl Taylor) at Logica receive our particular thanks for their sterling work through all the drafts, reworkings and re-arrangings.

Finally, we welcome the opportunity of acknowledging all those whose work this book reports on; many of their names appear in the text and the bibliographies.

N. D. Birrell and M. A. Ould
London, November 1983