Software Configuration Management

Macmillan Computer Science Series

Consulting Editor

Professor F. H. Sumner, University of Manchester

S.T. Allworth, *Introduction to Real-Time Software Design*
Ian O. Angell, *A Practical Introduction to Computer Graphics*
G.M. Birtwistle, *Discrete Event Modelling on Simula*
T.B. Boffey, *Graph Theory in Operations Research*
Richard Bornat, *Understanding and Writing Compilers*
J.K. Buckle, *Software Configuration Management*
J.C. Cluley, *Interfacing to Microprocessors*
Robert Cole, *Computer Communications*
Derek Coleman, *A Structured Programming Approach to Data**
Andrew J.T. Colin, *Fundamentals of Computer Science*
Andrew J.T. Colin, *Programming and Problem-solving in Algol 68**
S.M. Deen, *Fundamentals of Data Base Systems**
P.M. Dew and K.R. James, *Introduction to Numerical Computation in Pascal*
K.C.E. Gee, *Introduction to Local Area Computer Networks*
J.B. Gosling, *Design of Arithmetic Units for Digital Computers*
Roger Hutton, *Fortran for Computers*
Roger Hutton, *Z80 Assembly Language Programming for Students*
Roland N. Ibbett, *The Architecture of High Performance Computers*
Patrick Jaulent, *The 68000 – Hardware and Software*
H. Kopetz, *Software Reliability*
E.V. Krishnamurthy, *Introductory Theory of Computer Science*
Graham Lee, *From Hardware to Software: an introduction to computers*
A.M. Lister, *Fundamentals of Operating Systems, third edition**
G.P. McKeown and V.J. Rayward-Smith, *Mathematics for Computing*
Brian Meek, *Fortran, PL/I and the Algols*
Derrick Morris, *System Programming Based on the PDP11*
Christian Queinnec, *LISP*
John Race, *Case Studies in Systems Analysis*
W.P. Salaman, O. Tisserand and B. Toulout, *FORTH*
L.E. Scales, *Introduction to Non-linear Optimization*
Colin J. Theaker and Graham R. Brookes, *A Practical Course on Operating Systems*
M.J. Usher, *Information Theory for Information Technologists*
B.S. Walker, *Understanding Microprocessors*
Peter J.L. Wallis, *Portable Programming*
I.R. Wilson and A.M. Addyman, *A Practical Introduction to Pascal – with BS6192, second edition*

*The titles marked with an asterisk were prepared during the Consulting Editorship of Professor J.S. Rohl, University of Western Australia.

Software Configuration Management

J. K. Buckle, M.A., F.B.C.S.

© J. K. Buckle 1982

All rights reserved. No reproduction, copy or transmission of this publication may be made without written permission.

No paragraph of this publication may be reproduced, copied or transmitted save with written permission or in accordance with the provisions of the Copyright Act 1956 (as amended).

Any person who does any unauthorised act in relation to this publication may be liable to criminal prosecution and civil claims for damages.

First published 1982

Reprinted 1985

Published by
MACMILLAN EDUCATION LTD
Houndmills, Basingstoke, Hampshire RG21 2XS
and London
Companies and representatives
throughout the world

ISBN 978-0-333-33228-3

ISBN 978-1-349-16819-4 (eBook)

DOI 10.1007/978-1-349-16819-4

to my parents

Contents

<i>Preface</i>	xi
1 Basic Configuration Management Concepts	1
1.1 What is CM?	1
1.2 Baselines	2
1.3 Interaction with other control systems	2
1.4 Basic development controls	4
1.5 Post-development controls	6
1.6 The structure of the book	6
2 Configuration Management Mechanisms	9
2.1 The development process model	9
2.2 CM implications	15
2.3 Procedure publication	17
2.4 Project manager's checklist	18
2.5 Customisation for particular projects	20
3 Baselines	21
3.1 Baselines for software projects	23
3.2 Establishing development baselines	28
3.3 Specification formats and standards	31
3.4 Application to smaller projects	31
4 Configuration Identification	33
4.1 Establishing control units	34
4.2 Versions, forms and media	35
4.3 Item labelling	36
4.4 Registration	37
4.5 Practical application of CI	39
5 Document Production and Document Change Control	40
5.1 Document references and status	41

5.2 Producing specification documents	45
5.3 Changing red-status documents	48
5.4 Application to smaller projects	48
6 Code Change Control	52
6.1 Specification changes	52
6.2 Internal software changes	53
6.3 Baseline changes	53
6.4 Change control forms and systems	55
6.5 Application to smaller projects	55
7 Reviews and Inspections	57
7.1 Reviews	57
7.2 Inspections	61
7.3 Form of inspections	64
7.4 Application to smaller projects	65
8 Testing and validation	66
8.1 Development testing	66
8.2 Handover validation	67
8.3 Test libraries	68
8.4 Application to smaller projects	68
9 Internal Documentation	70
9.1 Document standards	70
9.2 Program documentation	71
9.3 Application to smaller projects	75
10 Maintenance and Production	76
10.1 The production process	77
10.2 CM implications	80
10.3 Software production baselines	82
10.4 Documentary formats and standards	84
10.5 Control of basis software	86
10.6 Control of delivered systems	87
10.7 Production quality assurance	90
10.8 Application to simple maintenance	90
11 CM Tools	92
11.1 Forms and standards	95
11.2 Archives	96

<i>Contents</i>	ix
11.3 Development and production systems	99
11.4 Practical application	103
12 Organisation for CM	105
12.1 Project initiation	105
12.2 Builds	107
12.3 Development handover	108
12.4 Error handling	109
12.5 Error correction	112
12.6 Organisation of staff	112
12.7 Introduction of CM	114
12.8 CM problems and pitfalls	116
<i>Appendix A Development CM Checklist</i>	119
<i>Appendix B Sample CM Procedures</i>	123
CM1 Development specification documents	124
CM2 Development document creation and change control	127
CM3 Development documents	133
CM4 Design inspections	140
CM5 Basis code and documentation change control	143
<i>References and Bibliography</i>	146
<i>Index</i>	149

Preface

Configuration management is a collection of techniques designed to improve the quality of software products by providing greater visibility, evidence of progress during development and production, and improved technical and management control. The term was originally applied to hardware development and production controls. Over the brief history of software systems the lack of such a control methodology has led to a number of well-publicised and costly disasters and it is in response to this that modern software configuration management has arisen. Many of the techniques are taken from the hardware experience, where a longer history has allowed good working methods to evolve; others have had to be developed from scratch in recognition of some of the unique properties and problems of software. Many are in fact good working practices that individual programmers apply to their own work without thinking, but that have needed to be extended and formalised to work with complex projects involving large numbers of people.

The application of configuration management, or CM, to software was pioneered by large computer manufacturers and specialised software teams working on large contracts, particularly in the aerospace and defence areas. As such, the procedures evolved tended to be relatively complex and comprehensive (TRW, for example, one of the leaders in the field, have a configuration management and software quality handbook some two inches thick [1]). Such complexity is of course entirely unsuitable for the average DP department project and, in any case, the environment in which work is commissioned in a DP department is totally different from that in which a major system contractor operates. Nevertheless, many of the fundamental ideas of CM are directly applicable to all software projects, providing they are suitably adapted to the different working methods. The author has practical experience of applying the methodology described in this book to situations as diverse as a major mainframe manufacturer's basic software, a special purpose mini-manufacturer's application software, the (large) DP department of a multi-national company and the four-man DP department of a sales company. In all cases there was a noticeable improvement in effectiveness and quality, which was actually more easily measurable in the smaller cases than the large ones. It is on this practical experience that the recommendations of the book are based.

The basic techniques described are equally relevant to: application software or systems software; commercial, scientific or process control; systems produced by computer users, manufacturers or software and systems houses. The way in which they will be applied will of course differ slightly, and the book is structured with this in mind.

The first two chapters of the book describe the overall aims and methods of configuration management and show its relevance and application in the areas mentioned above. Between them they establish the framework in which the individual techniques described in the rest of the book must be applied. Each of the remaining chapters describes an individual technique and is relatively free-standing. In total the mechanisms described will provide the methodology outlined in the first two chapters, but they are presented in the form of ideas, reasons and examples rather than as concrete procedures that all wishing to practise configuration management should obey. Examples are provided where possible but, within the limitations of a book this size, it is difficult to provide detailed case histories that are of general interest without giving also immense details of the background to CM operations in the organisations concerned.

Some guidance is, however, required on exactly how the principles of these chapters can be applied in practice. Appendix sections are therefore provided, each containing a skeleton procedure for the introduction of a particular control described in the main book. Each is based on an actual example that has been successfully operated by the author in one or more projects and is suitable for adoption by a project manager or group manager where it is agreed that the corresponding control is needed. Cross-references are provided to link the procedures to the relevant chapters of the main book, and indications are given of what modifications might be needed in particular circumstances. There is not a one-to-one correspondence between chapters and appendices, both since some techniques require several procedures and because the implementation of some of the configuration-management mechanisms is so much constrained by local circumstances that no general model is possible.

The book can thus be used in several ways

- as a method of updating the reader with modern software configuration management practice;
- as a source of ideas for introducing configuration management practices into existing control techniques;
- as a basis for building up a set of control procedures from scratch.

While, as stated above, the author has personal, practical experience of the successful application of each of the techniques described, he would be very interested to hear of readers' experience in implementing them in their own environments.

A selected bibliography on configuration management and associated matters is given at the end of the book for the reader who wishes to find out more about the history and practice of CM in software development.

Finally, three apologies. First, throughout the book the masculine pronouns and possessives are used for programmers, managers, etc. This is merely a shorthand for the more accurate, but much more clumsy he/she, him/her and his/hers. (Perhaps we should follow the Finns and adopt a single, sexless pronoun!) Second, although the history outlined above shows how the term 'configuration management' came to be applied to software, it is nevertheless in the author's opinion a highly confusing name. No doubt the reader could think of several better expressions for the control techniques described in this book. Unfortunately, particularly in the United States, the term has passed into common usage and to coin new names would only add to the confusion. Readers who, like the author, are upset by misuse of the language are advised to use the abbreviation CM and to let it stand for anything (control methodology?) or nothing as they think fit. Last, methods of controlling staff involved in software development and maintenance are not the most scintillating subject. As a result a book describing them is unlikely to make good bedside reading. Nevertheless, some degree of detail is essential if sufficient information is to be provided to allow the actual implementation of the methodology. More casual readers are asked to make allowance for this and, if they find themselves at any time less than enthralled, to skip to the next main subject area.

J. K. BUCKLE

Acknowledgements

The author wishes to thank TRW and in particular Barry Boehm, Director of Software Research and Technology, for permission to use their baseline diagrams in chapter 3, and the unit development folder details in chapter 9.