Macmillan Computer Science Series

Consulting Editor:

Professor F.H. Sumner, University of Manchester

- A. Abdellatif, J. Le Bihan, M. Limame, *Oracle – A User's Guide*
I. O. Angell, *High-resolution Computer Graphics Using C*
I. O. Angell and G. Griffith, *High-resolution Computer Graphics Using FORTRAN 77*
I. O. Angell and G. Griffith, *High-resolution Computer Graphics Using Pascal*
C. Bamford and P. Curran, *Data Structures, Files and Databases, second edition*
P. Beynon-Davies, *Database Systems*
P. Beynon-Davies, *Information Systems Development, second edition*
G.M. Birtwistle, *Discrete Event Modelling on Simula*
Linda E.M. Brackenbury, *Design of VLSI Systems – A Practical Introduction*
Alan Bradley, *Peripherals for Computer Systems*
G.R. Brookes and A.J. Stewart, *Introduction to occam2 on the Transputer*
P.C. Capon and P.J. Jinks, *Compiler Engineering Using Pascal*
B. S. Chalk, *Computer Organisation and Architecture*
Robert Cole, *Computer Communications, second edition*
Eric Davalo and Patrick Naïm, *Neural Networks*
S.M. Deen, *Principles and Practice of Database Systems*
Joyce Duncan, Lesley Rackley and Alexandria Walker, *SSADM in Practice*
D. England *et al.*, *A Sun User's Guide, second edition*
Jean Ettinger, *Programming in C++*
J.S. Florentin, *Microprogrammed Systems Design*
A.B. Fontaine and F. Barrand, *80286 and 90386 Microprocessors*
Michel Gauthier, *Ada – A Professional Course*
M.G. Hartley, M. Healey and P.G. Depledge, *Mini and Microcomputer Systems*
J.A. Hewitt and R.J. Frank, *Software Engineering in Modula-2 – An Object-oriented Approach*
Patrick Jaulent, *The 68000 – Hardware and Software*
M.J. King and J.P. Pardoe, *Program Design Using JSP – A Practical Introduction, second edition*
Bernard Leguy, *Ada – A Programmer's Introduction*
M. Léonard, *Database Design Theory*
David Lightfoot, *Formal Specification Using Z*
A.M. Lister and R.D. Eager, *Fundamentals of Operating Systems, fifth edition*
Elizabeth Lynch, *Understanding SQL*
Tom Manns and Michael Coleman, *Software Quality Assurance, second edition*
G.P. McKeown and V.J. Rayward-Smith, *Mathematical Foundations for Computing*
B.A.E. Meekings, T.P. Kudrycki and M.D. Soren, *A book on C, third edition*
R.J. Mitchell, *C++ Object-oriented Programming*
R.J. Mitchell, *Microcomputer Systems Using the STE Bus*
R.J. Mitchell, *Modula-2 Applied*
Ian Pratt, *Artificial Intelligence*
Pham Thu Quang and C. Chartier-Kastler, *MERISE in Practice*
A.J. Pilavakis, *UNIX Workshop*
E.J. Redfern, *Introduction to Pascal for Computational Mathematics*
F.D. Rolland, *Programming with VDM*
A.G. Sutcliffe, *Human-Computer Interface Design, second edition*
C.J. Theaker and G.R. Brookes, *Concepts of Operating Systems*
M. Thorin, *Real-time Transaction Processing*

continued overleaf

A.J. Tyrell, *Eiffel Object-Oriented Programming*

Other titles

Ian O. Angell and Dimitrios Tsoubelis, *Advanced Graphics on VGA and XGA Cards Using Borland C++*

N. Frude, *A Guide to SPSS/PC+, second edition*

Peter Grossman, *Discrete Mathematics for Computing*

H. Harper and A. Meadows, *GNVQ Advanced Information Technology*

Percy Mett, *Introduction to Computing*

P.D. Picton, *Neural Networks*

Tony Royce, *COBOL – An Introduction*

Tony Royce, *Structured COBOL – An Introduction*

Tony Royce, *C Programming*

Software Quality Assurance

Tom Manns and Michael Coleman

Second Edition

© Thomas S. Manns and Michael J. Coleman 1988, 1996

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 provision of the Copyright, Designs and Patents Act 1988, or under the terms of any licence permitting limited copying issued by the Copyright Licensing Agency, 90 Tottenham Court Road, London W1P 9HE.

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

First edition 1988

Second edition 1996

Published by

MACMILLAN PRESS LTD

Houndmills, Basingstoke, Hampshire RG21 2XS

and London

Companies and representatives
throughout the world

ISBN 978-0-333-59861-0 ISBN 978-1-349-13285-0 (eBook)

DOI 10.1007/978-1-349-13285-0

A catalogue record for this book is available
from the British Library.

Contents

Preface	viii
1 Introduction	1
1.1 Problems of defining software quality	1
1.2 An overview of software quality assurance	12
1.3 Quality systems	17
1.4 Software quality costs	22
1.5 Implications of open systems	28
1.6 The motivation to undertake quality assurance activities	30
1.7 Exercises	32
2 High-level Process Models	34
2.1 Managerial aspects	34
2.2 The role and chronology of process models	36
2.3 Software development life cycle	38
2.3.1 The code and fix model	38
2.3.2 The traditional life-cycle model	39
2.3.3 Prototyping and the evolutionary model	48
2.3.4 Automatic programming and the transform model	49
2.3.5 Risk analysis and the spiral development model	51
2.4 The SEI process maturity model	55
2.5 Exercises	60
3 Low-level Process Models	63
3.1 The need for low-level process models	63
3.2 Process architectures	64
3.3 Process improvement	68
3.3.1 Defect prevention schemes	68
3.3.2 The software engineering process group	71
3.4 Documenting the process	72
3.5 Exercises	76
4 Software Project Planning and Control	77
4.1 Cost modelling, Cocomo and function point analysis	77
4.2 The software project plan	82
4.3 Managerial and technical reviews	88
4.4 Auditing and the role of audits	89
4.5 Exercises	92

5	Metrics for the Quality Manager	95
5.1	Basic concepts	95
5.2	Assumptions about the software development environment	99
5.3	Product metrics	101
5.4	Process metrics	109
5.5	Resource metrics	122
5.6	Exercises	122
6	Reviews, Inspections and Walkthroughs	126
6.1	Technical reviews	129
6.2	Inspections	135
6.3	Walkthroughs	138
6.4	Review topics and checklists	139
6.5	Exercises	141
7	Software Quality Assurance Plans	143
7.1	Role of the software quality assurance plan	143
7.2	Content of the software quality assurance plan	144
7.3	Exercises	152
8	Software Configuration Management	153
8.1	Managing the evolution of software	153
8.2	Software configuration management plan	159
8.3	Computer-aided software configuration management	162
8.4	Exercises	166
9	Requirements	168
9.1	Introduction	168
9.2	Objectives	168
9.3	Specification	170
9.4	Validation	173
9.6	Exercises	174
10	Software Design	176
10.1	Software design principles	176
10.2	Software design methodologies	177
10.3	Software design verification	178
10.4	Tools	179
10.5	Exercises	181
11	Code	183
11.1	Programming languages	183
11.2	Defensive programming techniques	187
11.3	Standards	190

Contents

vii

11.4	Program development tools	200
11.6	Exercises	205
12	Function and System Testing	207
12.1	Test planning	208
12.2	The testing process	211
12.3	Documenting the testing process	216
12.4	Testing tools	222
12.5	Exercises	223
Bibliography		225
Index		232

Preface to the Second Edition

This edition like its predecessor aims to provide a first level introduction to software quality assurance for software quality and software engineers. We hope that it will also be of benefit to anyone studying software quality or software development. This edition has been extensively revised and new chapters added to reflect the evolution of the subject since the first edition was written.

There has been a huge increase in interest in software quality assurance since the first edition was published and much more is now known about the subject and its problems. The subject is now treated in a much more sophisticated way creating a need to measure the previously unmeasured.

We feel that the potential benefits from the use of metrics in planning, monitoring and controlling the software development process as well as for the certification of the product are such that their use will soon become widespread. This feeling is reflected in the treatment of the subject given in this edition.

In addition to attempting to reflect this change in attitude to the subject this edition expands the treatment of the earlier edition to better cover the managerial role in planning both the project and the environment within which software production takes place.

The nature of the problems however has not changed, software is still a complex product. It has been described in many standards as 'all instructions and data which are input into a computer to cause it to function in any mode; it includes operating systems, supervisory systems, compilers and test routines as well as application programs'. It also includes 'the documents used to define and describe the program (including flow charts, network diagrams and program listings)' as well as 'specifications, test plans, test data, test results and user instructions'. This remains the most useful definition from a quality assurance point of view.

Quality assurance can be described as

'A planned and systematic pattern of all actions necessary to provide adequate confidence that the item or project conforms to established technical requirements.'

It should be intuitively obvious from these definitions that software quality assurance is difficult, and likely to remain so. The word 'planned' implies that a model of the software production process is needed to determine and schedule the software quality assurance activities which will provide the confidence in the software. The word 'systematic' means not random and implies that we know the relationship between our actions and the quality of the software. In many instances this is not the case except at a very superficial level.

Expected developments in software engineering will not necessarily make software quality assurance any easier.

The introduction of CASE tools and the long awaited introduction of IPSEs into the development process should reduce the defect introduction rate and should significantly improve the quality of software produced using them. However these tools must be validated and this will be a difficult task.

It is probable that customers will continue to demand the development of more and more complex applications which will be harder to quality assure.

Software is ubiquitous and people often behave as if they had complete confidence in it. Software failures can now impact upon many areas of our lives and are much less likely to be tolerated than in the past when software was used only by a small number of people who understood that they were using an inherently unreliable system. Software failures can now affect people who were unaware of the fact that they were using software. The responsibility of the software providers to provide software of a quality which justifies this level of public confidence leads to demands on software quality assurance programmes which are difficult to meet.

The consequences of software failures are perhaps most significant in the areas of safety critical systems. These demand the most stringent software quality assurance programs to provide the required degree of confidence.

New techniques, developments in hardware and price reductions have opened up the prospect of new and radically different software development methodologies being used to produce software. These need to be defined and implemented in such a way that software quality assurance can provide the required degree of confidence in the quality of the software produced by such new methodologies.

It is now generally recognised that management has a key role to play in the production of quality software. Management must create the environment in which production can take place. The production process including the software quality assurance activities must be defined and work must be done in accordance with the defining procedures. Audits must be held to determine compliance with these procedures.

Support functions such as configuration management must be in place and used. It is not an easy task to design a configuration management system which all staff recognise as enhancing global productivity but is essential.

The failure of management to properly plan and control the software development process is now felt to have been responsible for much of the poor quality software produced in the past. Although obviously a simplification there seems to be general agreement that in addition to not providing a proper environment for production management failed to ensure that the production process made the software visible as it was being developed. It is managements role to ensure that the software is made visible as it is developed in order that it can be properly controlled. If the true state of the development is not visible then it cannot be controlled.

The desire for visibility and control has resulted in considerable emphasis on measurement both of the software and the process used to produce it. The whole area of metrics and their use in software project management and their potential use for product certification is now very important and is reflected in the treatment given in this edition. We have concentrated on the metrics which we think are useful in practice rather than discussing the vast range of rather esoteric metrics which have been proposed in the academic literature.

It is important for management to plan and having planned it is essential for management to track the implementation of these plans. Software development requires the performance of many synchronised activities of different types the performance of each type of activity can be viewed as the implementation of a plan. For instance it is now realised that the testing process is effectively a project in its own right, requiring plans, schedules and resources. The same is true for such things as configuration management and the quality assurance activities.

A planned schedule of audits must be held as part of this tracking process to demonstrate compliance with plans and procedures. The output from the audit process is often an input into a review.

It is essential that reviews are held at appropriate points in the development process. There are two types of review those of the output from the production process, sometimes known as technical reviews, and those of the production process itself, sometimes known as managerial or progress reviews. Progress reviews are a major part of the tracking process.

A prerequisite for any planning process is the selection of a process model even if it is not known by that name. This model describes the activities which must be performed during the development and their sequencing. It is now clear that there needs to be a hierarchy of process models. It is in the nature of these models that the highest level is used to make explicit the managerial principles that underlie the production process and the lower levels are used to implement and control the development itself.

The attention given to the importance of defining and documenting the process naturally led to investigations of how the process could be improved. The SEI process maturity model proposes a logical progression through various stages of the development of the production process. Although the applicability of this model in the UK economic context has been questioned it is an influential model which should be referred to by anyone responsible for the production of software. Its usefulness lies in its ability to suggest what improvements to the production process should be made next.

There is increasing interest in the certification of quality management systems. This interest is not always driven by the desire to manufacture high quality software. Sometimes the desire for certification seems to be market driven in the sense that the ability to claim certification is seen as conferring a commercial advantage and managers want to put the structures in place to secure this without apparently caring about the quality of the product which emerges from the

system. The TickIT initiative which includes a certification scheme for the IT sector industries has made a significant contribution to the understanding of what is required to produce high quality software.

Like the first edition, this book falls into two broad parts. The first part deals with the planning processes necessary to assure the quality of the software and the second covers the development phases. During the development phases it is necessary to ensure that what was planned is actually taking place; that quality is being built into the software. This involves the measurement of various aspects of the product and the process and the comparison of the values obtained with those expected from the plans. It may of course be necessary to update the plans during the development to take account of the deviations from the original plans. This comparison of expected and actual values may suggest that it is necessary to introduce remedial actions in some cases. It may be necessary to rely upon evidence provided by audits to interpret the meaning of any variances.

The authors would like to acknowledge the assistance they have received from Malcolm Stewart of Macmillan Press, from the reviewers and from the many members of staff of the Ministry of Defence, especially from the Directorate General of Defence Quality Assurance and the DRA with whom they have discussed the topic.

This revision was undertaken by Tom Manns.