
Undergraduate Topics in Computer Science

Series editor

Ian Mackie

Advisory Boards

Samson Abramsky, University of Oxford, Oxford, UK

Karin Breitman, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, Brazil

Chris Hankin, Imperial College London, London, UK

Dexter Kozen, Cornell University, Ithaca, USA

Andrew Pitts, University of Cambridge, Cambridge, UK

Hanne Riis Nielson, Technical University of Denmark, Kongens Lyngby, Denmark

Steven Skiena, Stony Brook University, Stony Brook, USA

Iain Stewart, University of Durham, Durham, UK

Undergraduate Topics in Computer Science (UTiCS) delivers high-quality instructional content for undergraduates studying in all areas of computing and information science. From core foundational and theoretical material to final-year topics and applications, UTiCS books take a fresh, concise, and modern approach and are ideal for self-study or for a one- or two-semester course. The texts are all authored by established experts in their fields, reviewed by an international advisory board, and contain numerous examples and problems. Many include fully worked solutions.

More information about this series at <http://www.springer.com/series/7592>

Gerard O'Regan

Concise Guide to Software Engineering

From Fundamentals to Application Methods

Gerard O'Regan
SQC Consulting
Cork
Ireland

ISSN 1863-7310 ISSN 2197-1781 (electronic)
Undergraduate Topics in Computer Science
ISBN 978-3-319-57749-4 ISBN 978-3-319-57750-0 (eBook)
DOI 10.1007/978-3-319-57750-0

Library of Congress Control Number: 2017939621

© Springer International Publishing AG 2017

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

In memory of my dear godmother

Mrs. Maureen Barry

Preface

Overview

The objective of this book was to provide a concise introduction to the software engineering field to students and practitioners. The principles of software engineering are discussed, and the goal is to give the reader a grasp of the fundamentals of the software engineering field, as well as guidance on how to apply the theory in an industrial environment.

Organization and Features

Chapter 1 presents a broad overview of software engineering, and discusses various software lifecycles and the phases in software development. We discuss requirements gathering and specification, software design, implementation, testing and maintenance. The lightweight Agile methodology is introduced, and it has become very popular in industry.

Chapter 2 provides an introduction to project management for traditional software engineering, and we discuss project estimation, project planning and scheduling, project monitoring and control, risk management, managing communication and change, and managing project quality.

Chapter 3 discusses requirements engineering and discusses activities such as requirements gathering, requirements elicitation, requirements analysis, requirements management, and requirements verification and validation.

Chapter 4 discusses design and development, and software design is the blueprint of the solution to be developed. It is concerned with the high-level architecture of the system, as well as the detailed design that describes the algorithms and functionality of the individual programmes. The detailed design is then implemented in a programming language such as C++ or Java. We discuss software development topics such as software reuse, customized-off-the-shelf software (COTS) and open-source software development.

Chapter 5 discusses software configuration management and discusses the fundamental concept of a baseline. Configuration management is concerned with identifying those deliverables that must be subject to change control, and controlling changes to them.

Chapter 6 discusses software inspections, which play an important role in building quality into a product. The well-known Fagan inspection process that was developed at IBM in the 1970s is discussed, as well as lighter review and walk-through methodologies.

Chapter 7 is concerned with software testing, and discusses the various types of testing that may be carried out during the project. We discuss test planning, test case definition, test environment set-up, test execution, test tracking, test metrics, test reporting and testing in an e-commerce environment.

Chapter 8 is concerned with the selection and management of a software supplier. It discusses how candidate suppliers may be identified, formally evaluated against defined selection criteria, and how the appropriate supplier is selected. We discuss how the selected supplier is managed during the project.

Chapter 9 discusses software quality assurance and the importance of process quality. It is a premise in the quality field that good processes and conformance to them is essential for the delivery of high-quality product, and this chapter discusses audits and describes how they are carried out.

Chapter 10 is concerned with software metrics and problem-solving, and this includes a discussion of the balanced score card which assists in identifying appropriate metrics for the organization. The Goal Question Metric (GQM) approach is discussed, and this allows appropriate metrics related to the organization goals to be defined. A selection of sample metrics for an organization is presented, and problem-solving tools such as fishbone diagrams, pareto charts and trend charts are discussed.

Chapter 11 discusses software reliability and dependability, and covers topics such as software reliability and software reliability models; the Cleanroom methodology, system availability; safety and security critical systems; and dependability engineering.

Chapter 12 discusses formal methods, which consist of a set of mathematical techniques to specify and derive a programme from its specification. Formal methods may be employed to rigorously state the requirements of the proposed system. They may be employed to derive a programme from its mathematical specification, and they may be used to provide a rigorous proof that the implemented programme satisfies its specification. They have been mainly applied to the safety critical field.

Chapter 13 presents the Z specification language, which is one of the more popular formal methods. It was developed at the Programming Research Group at Oxford University in the early 1980s. Z specifications are mathematical, and the use of mathematics ensures precision and allows inconsistencies and gaps in the specification to be identified. Theorem provers may be employed to demonstrate that the software implementation meets its specification.

Chapter 14 presents the unified modelling language (UML), which is a visual modelling language for software systems, and I used to present several views of the system architecture. It was developed at Rational Corporation as a notation for modelling object-oriented systems. We present various UML diagrams such as use case diagrams, sequence diagrams and activity diagrams.

Chapter 15 discusses software process improvement. It begins with a discussion of a software process, and discusses the benefits that may be gained from a software process improvement initiative. Various models that support software process improvement are discussed, and these include the Capability Maturity Model Integration (CMMI), ISO 9000, Personal Software Process (PSP) and Team Software Process (TSP).

Chapter 16 gives an overview of the CMMI model and discusses its five maturity levels and their constituent process areas. We discuss both the staged and continuous representations of the CMMI, and SCAMPI appraisals that indicate the extent to which the CMMI has been implemented in the organization, as well as identifying opportunities for improvement.

Chapter 17 discusses various tools to support the various software engineering activities. The focus is first to define the process and then to find tools to support the process. Tools to support project management are discussed as well as tools to support requirements engineering, configuration management, design and development activities and software testing.

Chapter 18 discusses the Agile methodology which is a popular lightweight approach to software development. Agile provides opportunities to assess the direction of a project throughout the development lifecycle, and ongoing changes to requirements are considered normal in the Agile world. It has a strong collaborative style of working, and it advocates adaptive planning and evolutionary development.

Chapter 19 discusses innovation in the software field including miscellaneous topics such as distributed systems, service-oriented architecture, software as a service, cloud computing and embedded systems. We discuss the need for innovation in software engineering, and discuss some recent innovations such as aspect-oriented software engineering.

Chapter 20 is the concluding chapter in which we summarize the journey that we have travelled in this book.

Audience

The main audience of this book are computer science students who are interested in learning about software engineering and in learning on how to build high-quality and reliable software on time and on budget. It will also be of interest to industrialists including software engineers, quality professionals and software managers, as well as the motivated general reader.

Acknowledgements

I am deeply indebted to family and friends who supported my efforts in this endeavour, and my thanks, as always, to the team at Springer. This book is dedicated to my late godmother (Mrs. Maureen Barry), who I always enjoyed visiting in Ringaskiddy, Co. Cork.

Cork, Ireland

Gerard O'Regan

Contents

1	Background	1
1.1	Introduction	1
1.2	What Is Software Engineering?	4
1.3	Challenges in Software Engineering	7
1.4	Software Processes and Lifecycles	8
1.4.1	Waterfall Lifecycle	9
1.4.2	Spiral Lifecycles	10
1.4.3	Rational Unified Process	11
1.4.4	Agile Development	12
1.5	Activities in Waterfall Lifecycle	15
1.5.1	User Requirements Definition	15
1.5.2	Specification of System Requirements	16
1.5.3	Design	16
1.5.4	Implementation	17
1.5.5	Software Testing	18
1.5.6	Support and Maintenance	19
1.6	Software Inspections	20
1.7	Software Project Management	21
1.8	CMMI Maturity Model	22
1.9	Formal Methods	23
1.10	Review Questions	24
1.11	Summary	24
	References	25
2	Software Project Management	27
2.1	Introduction	27
2.2	Project Start-up and Initiation	29
2.3	Estimation	30
2.3.1	Estimation Techniques	31
2.3.2	Work-Breakdown Structure	31
2.4	Project Planning and Scheduling	32
2.5	Risk Management	36
2.6	Quality Management in Projects	36
2.7	Project Monitoring and Control	38

2.8	Managing Issues and Change Requests	39
2.9	Project Board and Governance	40
2.10	Project Reporting	42
2.11	Project Closure	42
2.12	Prince 2 Methodology.	43
2.13	Review Questions.	45
2.14	Summary	45
	Reference	46
3	Requirements Engineering	47
3.1	Introduction	47
3.2	Requirements Process	48
3.2.1	Requirements Elicitation and Specification.	51
3.2.2	Requirements Analysis	54
3.2.3	Requirements Verification and Validation	54
3.2.4	Requirements Managements.	55
3.2.5	Requirements Traceability	56
3.3	System Modelling	57
3.4	Review Questions.	59
3.5	Summary	59
	References	60
4	Software Design and Development	61
4.1	Introduction	61
4.2	Architecture Design	62
4.3	Detailed Design and Development	66
4.3.1	Function-Oriented Design	67
4.3.2	Object-Oriented Design	67
4.3.3	User Interface Design	68
4.3.4	Open-Source Development	70
4.3.5	Customized Off-the-Shelf Software	70
4.3.6	Software Reuse	71
4.3.7	Object-Oriented Programming	71
4.4	Software Maintenance and Evolution	73
4.5	Review Questions.	73
4.6	Summary	73
	References	74
5	Configuration Management	75
5.1	Introduction	75
5.2	Configuration Management System.	79
5.2.1	Identify Configuration Items	80
5.2.2	Document Control Management	80
5.2.3	Source Code Control Management	81
5.2.4	Configuration Management Plan.	81
5.3	Change Control	82

5.4	Configuration Management Audits	84
5.5	Review Questions.	85
5.6	Summary	86
6	Software Inspections	87
6.1	Introduction	87
6.2	Economic Benefits of Software Inspections	89
6.3	Informal Reviews	90
6.4	Structured Walk-through	91
6.5	Semi-formal Review Meeting.	91
6.6	Fagan Inspections.	92
6.6.1	Fagan Inspection Guidelines	93
6.6.2	Inspectors and Roles.	96
6.6.3	Inspection Entry Criteria	96
6.6.4	Preparation	96
6.6.5	The Inspection Meeting.	98
6.6.6	Inspection Exit Criteria	99
6.6.7	Issue Severity	99
6.6.8	Defect Type.	100
6.7	Automated Software Inspections.	101
6.8	Review Questions.	103
6.9	Summary	104
	References	104
7	Software Testing	105
7.1	Introduction	105
7.2	Test Process	107
7.3	Test Planning.	111
7.4	Test Case Design and Definition	112
7.5	Test Execution.	113
7.6	Test Reporting and Project Sign-Off	114
7.7	Testing and Quality Improvement.	115
7.8	Traceability of Requirements	116
7.9	Test Tools.	116
7.9.1	Test Management Tools	116
7.9.2	Miscellaneous Testing Tools	117
7.10	e-Commerce Testing.	118
7.11	Test-Driven Development	119
7.12	Review Questions.	120
7.13	Summary	121
8	Supplier Selection and Management	123
8.1	Introduction	123
8.2	Planning and Requirements	125
8.3	Identifying Suppliers.	125
8.4	Prepare and Issue RFP	126

8.5	Evaluate Proposals and Select Supplier	126
8.6	Formal Agreement	127
8.7	Managing the Supplier	128
8.8	Acceptance of Software.	128
8.9	Roll-out and Customer Support	129
8.10	Review Questions.	129
8.11	Summary	129
9	Software Quality Assurance	131
9.1	Introduction	131
9.2	Audit Planning.	134
9.3	Audit Meeting	135
9.4	Audit Reporting	136
9.5	Follow-Up Activity.	136
9.6	Audit Escalation.	137
9.7	Review of Audit Activities	137
9.8	Other Audits	137
9.9	Review Questions.	138
9.10	Summary	138
10	Software Metrics and Problem-Solving	139
10.1	Introduction	139
10.2	The Goal, Question, Metric Paradigm	141
10.3	The Balanced Scorecard	143
10.4	Metrics for an Organization	145
10.4.1	Customer Satisfaction Metrics	145
10.4.2	Process Improvement Metrics.	146
10.4.3	Human Resources and Training Metrics	148
10.4.4	Project Management Metrics	149
10.4.5	Development Quality Metrics.	151
10.4.6	Quality Audit Metrics	153
10.4.7	Customer Care Metrics	155
10.4.8	Miscellaneous Metrics.	157
10.5	Implementing a Metrics Programme	159
10.5.1	Data Gathering for Metrics	160
10.6	Problem-Solving Techniques	161
10.6.1	Fishbone Diagram	162
10.6.2	Histograms	164
10.6.3	Pareto Chart	165
10.6.4	Trend Graphs.	166
10.6.5	Scatter Graphs	167
10.6.6	Metrics and Statistical Process Control	168
10.7	Review Questions.	169
10.8	Summary	169
	References	170

11	Software Reliability and Dependability	171
11.1	Introduction	171
11.2	Software Reliability	172
11.2.1	Software Reliability and Defects	173
11.2.2	Cleanroom Methodology	175
11.2.3	Software Reliability Models	176
11.3	Dependability	178
11.4	Computer Security	180
11.5	System Availability	181
11.6	Safety Critical Systems	182
11.7	Review Questions	183
11.8	Summary	184
	References	184
12	Formal Methods	185
12.1	Introduction	185
12.2	Why Should We Use Formal Methods?	187
12.3	Applications of Formal Methods	189
12.4	Tools for Formal Methods	190
12.5	Approaches to Formal Methods	190
12.5.1	Model-Oriented Approach	190
12.5.2	Axiomatic Approach	192
12.6	Proof and Formal Methods	193
12.7	The Future of Formal Methods	194
12.8	The Vienna Development Method	194
12.9	VDM ⁺ , The Irish School of VDM	196
12.10	The Z Specification Language	197
12.11	The B-Method	198
12.12	Predicate Transformers and Weakest Preconditions	199
12.13	The Process Calculi	200
12.14	Finite State Machines	200
12.15	The Parnas Way	201
12.16	Usability of Formal Methods	202
12.17	Review Questions	205
12.18	Summary	205
	References	206
13	Z Formal Specification Language	209
13.1	Introduction	209
13.2	Sets	212
13.3	Relations	213
13.4	Functions	215
13.5	Sequences	216
13.6	Bags	217
13.7	Schemas and Schema Composition	218

13.8	Reification and Decomposition	221
13.9	Proof in Z	222
13.10	Review Questions	223
13.11	Summary	223
	References	224
14	Unified Modelling Language	225
14.1	Introduction	225
14.2	Overview of UML	226
14.3	UML Diagrams	228
14.4	Object Constraint Language	234
14.5	Tools for UML	235
14.6	Rational Unified Process	235
14.7	Review Questions	237
14.8	Summary	238
	References	238
15	Software Process Improvement	239
15.1	Introduction	239
15.2	What Is a Software Process?	240
15.3	What Is Software Process Improvement?	242
15.4	Benefits of Software Process Improvement	243
15.5	Software Process Improvement Models	244
15.6	Process Mapping	247
15.7	Process Improvement Initiatives	248
15.8	Barriers to Success	249
15.9	Setting Up an Improvement Initiative	249
15.10	Appraisals	251
15.11	Review Questions	253
15.12	Summary	253
	References	254
16	Capability Maturity Model Integration	255
16.1	Introduction	255
16.2	The CMMI	258
16.3	CMMI Maturity Levels	261
	16.3.1 CMMI Representations	264
16.4	Categories of CMMI Processes	266
16.5	CMMI Process Areas	267
16.6	Components of CMMI Process Areas	270
16.7	SCAMPI Appraisals	275
16.8	Review Questions	275
16.9	Summary	276
	References	276

17	Software Engineering Tools	279
17.1	Introduction	279
17.2	Tools for Project Management	280
17.3	Tools for Requirements	284
17.4	Tools for Design and Development.	287
17.5	Tools for Configuration Management and Change Control.	290
17.6	Tools for Code Analysis and Code Inspections	290
17.7	Tools for Testing	292
17.8	Review Questions.	294
17.9	Summary	294
	Reference	295
18	Agile Methodology	297
18.1	Introduction	297
18.2	Scrum Methodology	300
18.3	User Stories.	301
18.4	Estimation in Agile.	302
18.5	Test-Driven Development	302
18.6	Pair Programming	303
18.7	Review Questions.	304
18.8	Summary	304
	Reference	305
19	A Miscellany of Innovation	307
19.1	Introduction	307
19.2	Distributed Systems	308
19.3	Service-Oriented Architecture.	309
19.4	Software as a Service	310
19.5	Cloud Computing.	311
19.6	Embedded Systems.	312
19.7	Software Engineering and Innovation	313
19.7.1	Aspect-Oriented Software Engineering	313
19.8	Review Questions.	314
19.9	Summary	314
	References	315
20	Epilogue	317
20.1	The Future of Software Engineering	319
	Glossary	321
	References	327
	Index	329

List of Figures

Fig. 1.1	Standish report—results of 1995 and 2009 survey	3
Fig. 1.2	Standish 1998 report—estimation accuracy	7
Fig. 1.3	Waterfall V lifecycle model	9
Fig. 1.4	SPIRAL lifecycle model ... public domain	10
Fig. 1.5	Rational Unified Process	12
Fig. 2.1	Simple process map for project planning	34
Fig. 2.2	Sample microsoft project schedule	34
Fig. 2.3	Simple process map for project monitoring and control	39
Fig. 2.4	Prince 2 project board	41
Fig. 2.5	Project management triangle	43
Fig. 2.6	Prince 2 processes	44
Fig. 3.1	Requirements process	52
Fig. 4.1	C.A.R Hoare (public domain)	64
Fig. 4.2	David Parnas	65
Fig. 5.1	Simple process map for change requests	83
Fig. 5.2	Simple process map for configuration management	84
Fig. 6.1	Michael Fagan	88
Fig. 6.2	Template for semi-formal review	94
Fig. 6.3	Sample defect types in a project (ODC)	101
Fig. 6.4	Template for Fagan inspection	102
Fig. 7.1	Simplified test process	108
Fig. 7.2	Sample test status	110
Fig. 7.3	Cumulative defects	114
Fig. 9.1	Sample audit process	133
Fig. 10.1	GQM example	141
Fig. 10.2	The balanced scorecard	143
Fig. 10.3	Balanced score card and implementing strategy	143
Fig. 10.4	Customer survey arrivals	145
Fig. 10.5	Customer satisfaction measurements	146
Fig. 10.6	Process improvement measurements	146
Fig. 10.7	Status of process improvement suggestions	147
Fig. 10.8	Age of open process improvement suggestions	147
Fig. 10.9	Process improvement productivity	148
Fig. 10.10	Employee headcount in current year	148

Fig. 10.11	Employee turnover in current year	149
Fig. 10.12	Schedule timeliness metric	149
Fig. 10.13	Effort timeliness metric	150
Fig. 10.14	Requirements delivered	151
Fig. 10.15	Total number of issues in project	151
Fig. 10.16	Open issues in project	152
Fig. 10.17	Age of open defects in project	152
Fig. 10.18	Problem arrivals per month	153
Fig. 10.19	Phase containment effectiveness	153
Fig. 10.20	Annual audit schedule	154
Fig. 10.21	Status of audit actions	154
Fig. 10.22	Audit action types	155
Fig. 10.23	Customer queries (arrivals/closures)	155
Fig. 10.24	Outage time per customer	156
Fig. 10.25	Availability of system per month	157
Fig. 10.26	Configuration management	157
Fig. 10.27	CMMI maturity in current year	158
Fig. 10.28	Cost of poor quality (COPQ)	159
Fig. 10.29	Fishbone cause-and-effect diagram	163
Fig. 10.30	Histogram	165
Fig. 10.31	Pareto chart outages	166
Fig. 10.32	Trend chart estimation accuracy	167
Fig. 10.33	Scatter graph amount inspected rate/error density	168
Fig. 10.34	Estimation accuracy and control charts	168
Fig. 12.1	Deterministic finite state machine	202
Fig. 13.1	Specification of positive square root	210
Fig. 13.2	Specification of a library system	212
Fig. 13.3	Specification of borrow operation	212
Fig. 13.4	Specification of vending machine using bags	218
Fig. 13.5	Schema inclusion	220
Fig. 13.6	Merging schemas ($S_1 \vee S_2$)	220
Fig. 13.7	Schema composition	221
Fig. 13.8	Refinement commuting diagram	222
Fig. 14.1	Simple object diagram	230
Fig. 14.2	Use case diagram of ATM machine	231
Fig. 14.3	UML sequence diagram for balance enquiry	232
Fig. 14.4	UML activity diagram	233
Fig. 14.5	UML state diagram	233
Fig. 14.6	Iteration in rational unified process	236
Fig. 14.7	Phases and workflows in rational unified process	237
Fig. 15.1	Process as glue for people, procedures and tools	241
Fig. 15.2	Sample process map	242
Fig. 15.3	Steps in process improvement	243
Fig. 15.4	ISO 9001 quality management system	246

Fig. 15.5	Continuous improvement cycle	250
Fig. 15.6	Appraisals	252
Fig. 16.1	Process as glue for people, procedures and tools	256
Fig. 16.2	ISO/IEC 12207 standard for software engineering processes	257
Fig. 16.3	CMMI worldwide maturity 2013	260
Fig. 16.4	CMMI maturity levels	264
Fig. 16.5	CMMI capability levels	265
Fig. 16.6	CMMI—continuous representation	265
Fig. 16.7	CMMI-staged model	270
Fig. 16.8	Specific practices for SG1—manage requirements	271
Fig. 17.1	Dashboard views in planview enterprise	283
Fig. 17.2	Planview process builder	284
Fig. 17.3	IBM Rational DOORS tool	286
Fig. 17.4	IBM Rational Software Modeler	287
Fig. 17.5	Sparx Enterprise Architect	288
Fig. 17.6	LDRA code coverage analysis report	291
Fig. 17.7	HP Quality Center	293
Fig. 19.1	A distributed system	308
Fig. 19.2	Service-oriented architecture	310
Fig. 19.3	Cloud computing. Creative commons	311
Fig. 19.4	Example of an embedded system	312

List of Tables

Table 2.1	Estimation techniques	32
Table 2.2	Example work-breakdown structure	33
Table 2.3	Sample project management checklist	35
Table 2.4	Risk management activities	37
Table 2.5	Activities in managing issues and change requests	40
Table 2.6	Project board roles and responsibilities	41
Table 2.7	Key processes in Prince 2	44
Table 3.1	Characteristics of good requirements	49
Table 3.2	Symptoms of poor requirements process	50
Table 3.3	Managing change requests	57
Table 3.4	Sample trace matrix	58
Table 4.1	Views of system architecture	63
Table 4.2	Object-oriented paradigm	69
Table 5.1	Features of good configuration management	76
Table 5.2	Symptoms of poor configuration management	77
Table 5.3	Software configuration management activities	78
Table 5.4	Build plan for project	78
Table 5.5	CMMI requirements for configuration management	79
Table 5.6	Sample configuration management audit checklist	85
Table 6.1	Informal review	90
Table 6.2	Structured walk-throughs	91
Table 6.3	Activities for semi-formal review meeting	93
Table 6.4	Overview Fagan inspection process	95
Table 6.5	Strict Fagan inspection guidelines	96
Table 6.6	Tailored (Relaxed) Fagan inspection guidelines	96
Table 6.7	Inspector roles	97
Table 6.8	Fagan entry criteria	97
Table 6.9	Inspection meeting	98
Table 6.10	Fagan exit criteria	99
Table 6.11	Issue severity	99
Table 6.12	Classification of defects in Fagan inspections	100
Table 6.13	Classification of ODC defect types	100
Table 7.1	Types of testing	109
Table 7.2	Simple test schedule	112

Table 8.1	Supplier selection and management	124
Table 9.1	Auditing activities	132
Table 9.2	Sample auditing checklist	135
Table 9.3	Sample audit report	136
Table 10.1	BSC objectives and measures for IT service organization	144
Table 10.2	Cost of quality categories	158
Table 10.3	Implementing metrics	159
Table 10.4	Goals and questions.	160
Table 10.5	Phase containment effectiveness	160
Table 11.1	Adam's 1984 study of software failures of IBM products	175
Table 11.2	New and old version of software.	175
Table 11.3	Cleanroom results in IBM	176
Table 11.4	Characteristics of good software reliability model	177
Table 11.5	Software reliability models	177
Table 11.6	Dimensions of dependability	180
Table 12.1	Criticisms of formal methods.	188
Table 12.2	Parnas's contributions to software engineering	202
Table 12.3	Techniques for validation of formal specification.	204
Table 12.4	Why are formal methods difficult?.	204
Table 12.5	Characteristics of a usable formal method	204
Table 13.1	Schema composition	220
Table 14.1	Classification of UML things.	227
Table 14.2	UML diagrams	228
Table 14.3	Simple class diagram.	229
Table 14.4	Advantages of UML	234
Table 14.5	OCL constraints.	235
Table 14.6	UML tools.	235
Table 15.1	Benefits of software process improvement (CMMI).	245
Table 15.2	Continuous improvement cycle	251
Table 15.3	Teams in improvement programme	252
Table 15.4	Phases in an Appraisal.	253
Table 16.1	Motivation for CMMI implementation.	260
Table 16.2	Benefits of CMMI implementation	261
Table 16.3	CMMI maturity levels	262
Table 16.4	CMMI capability levels for continuous representation.	266
Table 16.5	CMMI process categories	267
Table 16.6	CMMI Process Areas.	268
Table 16.7	CMMI generic practices	272
Table 16.8	Implementation of generic practices.	274
Table 17.1	Tool evaluation table.	280
Table 17.2	Key capabilities of planview enterprise	283
Table 17.3	Tools for requirements development and management.	285
Table 17.4	Tools for software design	287
Table 17.5	Integrated development environment	289