

# Software Quality Assurance

Integrating Testing, Security, and Audit

Abu Sayed Mahfuz



CRC Press

Taylor & Francis Group

Boca Raton London New York

---

CRC Press is an imprint of the  
Taylor & Francis Group, an **informa** business  
AN AUERBACH BOOK

Software Quality Assurance: Integrating Testing, Security, and Audit  
<https://www.crcpress.com/9781498735537>  
for IT Business Edge

CRC Press  
Taylor & Francis Group  
6000 Broken Sound Parkway NW, Suite 300  
Boca Raton, FL 33487-2742

© 2016 by Taylor & Francis Group, LLC  
CRC Press is an imprint of Taylor & Francis Group, an Informa business

No claim to original U.S. Government works

Printed on acid-free paper  
Version Date: 20151124

International Standard Book Number-13: 978-1-4987-3553-7 (Hardback)

This book contains information obtained from authentic and highly regarded sources. Reasonable efforts have been made to publish reliable data and information, but the author and publisher cannot assume responsibility for the validity of all materials or the consequences of their use. The authors and publishers have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission to publish in this form has not been obtained. If any copyright material has not been acknowledged please write and let us know so we may rectify in any future reprint.

Except as permitted under U.S. Copyright Law, no part of this book may be reprinted, reproduced, transmitted, or utilized in any form by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying, microfilming, and recording, or in any information storage or retrieval system, without written permission from the publishers.

For permission to photocopy or use material electronically from this work, please access [www.copyright.com](http://www.copyright.com) (<http://www.copyright.com/>) or contact the Copyright Clearance Center, Inc. (CCC), 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400. CCC is a not-for-profit organization that provides licenses and registration for a variety of users. For organizations that have been granted a photocopy license by the CCC, a separate system of payment has been arranged.

**Trademark Notice:** Product or corporate names may be trademarks or registered trademarks, and are used only for identification and explanation without intent to infringe.

---

#### Library of Congress Cataloging-in-Publication Data

---

Names: Mahfuz, Abu Sayed, author.  
Title: Software quality assurance : integrating testing, security, and audit / author, Abu Sayed Mahfuz.  
Description: Boca Raton : Taylor & Francis, 2015. | Includes bibliographical references and index.  
Identifiers: LCCN 2015045370 | ISBN 9781498735537 (alk. paper)  
Subjects: LCSH: Computer software--Quality control. | Computer software--Testing.  
Classification: LCC QA76.76.Q35 M34 2015 | DDC 005.3028/7--dc23  
LC record available at <http://lccn.loc.gov/2015045370>

---

Visit the Taylor & Francis Web site at  
<http://www.taylorandfrancis.com>

and the CRC Press Web site at  
<http://www.crcpress.com>

Software Quality Assurance: Integrating Testing, Security, and Audit  
<https://www.crcpress.com/9781498735537>  
for IT Business Edge

# Contents

<b>PREFACE</b>	xxv
<b>CONTENT OVERVIEW</b>	xxvii
<b>ACKNOWLEDGMENTS</b>	xxxi
<b>AUTHOR</b>	xxxiii

## **SECTION I CONCEPT**

<b>CHAPTER 1</b>	<b>QUALITY CONCEPT AND PERSPECTIVES</b>	3
	Introduction	3
	Part 1: Software Quality Concept	3
	Defining Software Quality	3
	Integrating Test, Security, and Audit	5
	Why Is Software Quality Important?	6
	What Is the Benefit of Software Quality in Business?	7
	Lack of Quality Is the Reason for Failure	8
	Failure Factors	8
	Part 2: Software Quality Characteristics	10
	What Is the Business Benefit of Quality Characteristics?	10
	Standard for Quality Characteristics ISO/IEC 9126	10
	Quality Characteristics	11
	Detailed Descriptions of Quality Characteristics	13
	Functionality	13
	Suitability	13
	Accuracy	13
	Interoperability	13
	Security	13
	Functionality Compliance	13

Reliability	13
Maturity	13
Fault Tolerance	13
Recoverability	13
Reliability Compliance	13
Usability	14
Understandability	14
Learnability	14
Operability	14
Usability Compliance	14
Efficiency	14
Time Behavior	14
Resource Behavior	14
Efficiency Compliance	14
Maintainability	14
Analyzability	14
Changeability	14
Stability	14
Testability	14
Maintainability Compliance	14
Portability	15
Adaptability	15
Installability	15
Coexistence/Conformance	15
Portability Compliance	15
Control Objectives for Information and Related Technology (COBIT)	15
Introduction	15
Meta-Requirements	16
Capability Maturity Model Integration (CMMI)	17
Quality Characteristics, COBIT, and CMMI	18
Part 3: Validation and Verification	18
Role of V&V in Software Quality	20
Software V&V Processes	20
V&V Task Reports	20
V&V Activity Summary Reports	21
V&V Anomaly Reports	21
Testing: Application	22
Unit Testing Plan	22
Determine Features to Be Tested	22
Design the Test Set	22
Implement the Test Plan	23
Execute the Test Procedures	23
Part 4: Reviews and Audit	23
Management Reviews	24
Application	25
Focus	25

Input	25
When to Conduct a Management Review	25
Review Procedures	26
Planning	26
Preparation and Execution	26
Technical Reviews	26
Responsibilities	27
Input	27
Inspections	27
Responsibilities	28
Inspection Rules and Procedures	29
Walkthroughs	29
Responsibilities	30
Audits	32
 <b>CHAPTER 2 MANAGEMENT AND PROCESS</b>	 35
Introduction	35
Part 1: Software Management	35
Software Management	35
Information Governance	35
Information Governance, IT Governance, and Data Governance	36
IT Governance	36
Data Governance	36
IG–EG and Strategic Planning	36
Making the Process Systematic	37
IT Process Alignment	38
The Expert Models for Software Management	38
ISO 12207/IEEE 12207.0	39
Serves as a Model for	39
Integration of IEEE 12207 and SESC	39
Acquisition	39
Development	39
Operation	40
Supporting Documentation	40
Part 2: Software Life Cycle Models	40
What Is Software Life Cycle?	40
Life Cycle Models	41
Boehm’s Spiral	41
Agile Methodology	41
What Is Agile? What Does It Mean?	42
Agile Principles	43
Waterfall	45
Part 3: Life Cycle Processes	45
Primary Life Cycle Process	46
Acquisition Process	47
Supply Process	48

Development Process	49
Operations Process	50
Maintenance Process	51
Supporting Life Cycle Processes	52
Documentation Process	52
Configuration Management Process	52
Quality Assurance Process	53
Verification Process	54
Validation Process	54
Joint Review Process	55
Audit Process	55
Audit Process Tasks	56

## SECTION II TESTING

<b>CHAPTER 3 TESTING: CONCEPT AND DEFINITION</b>	59
Introduction	59
Part 1: Testing in the Software Life Cycle	59
What Is Software Testing?	59
Requirements	60
Identification and Specification	60
Specification	60
Functional System Development	60
Technical System Design	61
Component Specification	61
Coding	61
Testing	61
Are We Building the Right System?	61
Are We Building the System Right?	62
Part 2: Software Testing Life Cycle	62
SDLC and STLC	63
Part 3: Kinds/Types of Testing	65
Black Box Testing	65
White Box Testing	65
Unit Testing	65
Integration Testing	66
Incremental Integration Testing	66
Functional Testing	66
System Testing	66
End-to-End Testing	67
Sanity Testing	67
Regression Testing	67
Acceptance Testing	68
Load Testing	68
Stress Testing	68
Performance Testing	69
Usability Testing	69
Install/Uninstall Testing	69

Recovery Testing	69
Security Testing	70
Comparison Testing	70
Alpha Testing	70
Beta Testing	70
Automated Testing	70
Agile Testing	71
Suggested Readings	71
<b>CHAPTER 4 TESTING: PLAN AND DESIGN</b>	73
Introduction	73
Part 1: Plan and Strategy	73
Test Plan	73
Contents of a Test Plan	73
Test Plan Identification	73
Document Change Control Log	74
Purpose of the Document	74
References	75
Sample Reference Metrics	75
Software Product Overview/Project Description	75
Test Objectives	75
Software Risk Issue and Mitigation	76
Communication and Status Reporting	76
Test Tools	77
Test Scope	77
Part 2: Test Approach and Stages	77
Requirements Analysis	77
Solution Specifications	78
Testing Levels	78
Unit Testing	78
System/Integration Testing	78
System Test	79
System Test Execution	79
Defect Management	79
Acceptance Testing	79
Test Data Preparation	80
Test Environments	80
Sample Entry/Exit Criteria	80
Test Schedule	81
Defect Reporting and Tracking	81
Roles and Responsibilities	83
Appendix	83
Reference Documents	83
Testing Estimation	84
Lessons Learned	84
Project Description	84
What Went Well	88
What Could Have Gone Better	88

NEW Opportunities	88
LOE Accuracy	88
Top Three Recommended Improvements	88
Part 3: Test Design Factors	88
Software Requirement	88
Requirement Identification	89
Requirement Identifier	89
Software Requirement Specification	90
Requirements Evaluation Matrix	91
Business Value of Requirements	93
Scales/Measures	93
Significant Requirement Conflicts and Enablers	93
Estimated Costs and Risks to Satisfy Requirements	93
Scales/Measures	94
Requirements Cost/Benefit and Prioritization Summary	94
Part 4: Test Case Specification and Design	95
Test Case Specification	95
Deliverables	95
Test Environment Setup	95
Deliverables	95
Sample Test Cases	96
Introduction	96
Scope	96
Objective	97
Sample Test Cases	97
Testing Condition 1.1—Login with Correct User ID and Password	97
Testing Condition 1.2—Wrong User ID	97
Testing Condition 1.3—Wrong Password	98
Testing Condition 1.4—Username Blank	98
Testing Condition 1.5—Password Blank	99
Testing Condition 1.6—Username and Password Blank	99
Testing Condition 1.7—Cancel Button Clicked	100
Testing Condition 1.8—Invalid User	100
Summary	101
<b>CHAPTER 5 TEST: EXECUTION AND REPORTING</b>	103
Introduction	103
Part 1: Starting Test Execution	103
Getting Ready to Start Test Execution	103
Requirement Coverage	104
Requirements Test Coverage Statement	105
Scheduling Test Runs	105
Assigning Test Execution	105
Part 2: Test Result Reporting	107
Status Report	107
Daily Stand-Up Update by Individual	107
Weekly Status Report Template	108



Test Result Summary Report	109
Document Change Control Log	109
Purpose of the Document	109
References: (Sample Reference Metrics)	109
Progression Test Case Execution Status	110
Regression Test Case Execution Status	111
Part 3: View and Analyze Test Results	112
Defect: As a Part of Test Result	112
Requirement Test Case—Defect Traceability	
Metrics	112
Defect Details	112
Deferred Defects	113
Defects by Root Cause	113
Canceled Defects	113
Defect Summary	113
Requirement Traceability Matrices (RTM)	115
System Test Coverage Metrics (Sample)	116
Test Execution Quality Metrics	116
Defect Tracking Overview	117
Defect Linkage	117

## **SECTION III CHALLENGES**

<b>CHAPTER 6 INCIDENT MANAGEMENT</b>	121
Introduction	121
Overview on Incident Management	121
Why Incident Management Is Important	122
Part 1: Identification	123
Definition	123
Incident	123
Information Security Incident	123
Accident	123
Defect	124
Failure	124
Incident Identification	124
Identifying Ways	124
Identifying the Attacking Hosts	125
Incident Initial Documentation	125
Incident Classification	126
Type of Incident	127
Initial Assessment	127
Part 2: Investigation and Analysis	128
Reasons to Investigate	128
Investigation Process	128
Incident Root Cause	129
Collecting Evidences	129
Six Steps for Successful Incident Investigation	130

Incident Analysis	130
Some Examples of Analyzing an Incident	131
Barrier Analysis	131
Damage Mode Effect Analysis	132
Scenario Analysis	132
Time/Loss Analysis for Emergence Response	
Evaluation	133
Analyzing Warning Time	133
Part 3: Response and Recovery	133
Incident Response	133
Initiate Recovery Mechanisms	136
Review Preliminary Investigation Results	136
Preventing Incidents	137
Incident Notification	137
Evidence Collection and Documentation	138
Part 4: Issues	139
Issues List	139
Project Issues List Instructions	139
Project Issues Log	141
Part 5: Security Incidents	141
Security Incidents Reporting	141
Before an Incident Happens the Team Should	141
After an Incident Happens	141
Responding to a Security Incident	142
Tips for Responding to Security Incidents	142
Steps to Take during the Incident	142
Responding to Security Violations	142
Security Office Actions	143
 <b>CHAPTER 7 DEFECT MANAGEMENT</b>	 145
Introduction	145
Part 1: Definition and Analysis	145
Definitions	145
Defect	145
Definition of an Error	146
Defect Repository	146
What Causes Defects in Software	146
Detecting a Defect Early	148
What Is the Cost of Defects Not Being Detected Early?	148
Defect Life Cycle Steps	150
Step 1: Recognition or Identification	150
Step 2: Investigation	150
Step 3: Action	150
Step 4: Disposition	151
Objectives of Testing	151
Reduce the Risk of Failure	151
Reduce the Cost of Testing	151

Analyze Root Causes	151
Address Causes of Defects	152
Institutionalize a Defined Process	152
Implement the Action Proposals	153
Part 2: Process and Methodology	153
Defect Management Process	153
Identifying	153
Categorizing	153
Prioritizing	153
Assigning	154
Resolving	154
Verifying	154
Closing	154
Management Reporting	154
Roles and Responsibilities in Software Development	
Life Cycle	154
Business Owner	154
Stakeholders	154
Analyst	155
Developer	155
Tester	155
Conflict Resolution and Escalations during Defect	155
Defect Management Methodology	156
Document Change Control	156
Documentation	156
Statement of Purpose	157
Risks	157
Defect Steps	157
Defect States	158
Defect Attributes	160
Defect Priorities	162
Defect Severities	162
Part 3: Root Cause Analysis	163
Definition	163
Root Cause Fields	163
Requirements	164
Defect Cause in Requirement	164
Incomplete/Missing	164
Inconsistent	164
Incorrect	164
Not Traceable	164
Not Testable	164
Implementation Dependent	164
Design	164
Code	164
Environment	165
Test	165

Data	165
Analysis	165
The Most Common Root Cause Classification	165
Defect Prevention	167
Benefits of Defect Prevention	167
Defect Prediction	170
 <b>CHAPTER 8 RISK, VULNERABILITY, AND THREAT</b>	
<b>MANAGEMENT</b>	171
Introduction	171
Part 1: Risk Management	171
Types of Risks	172
Impact of Risk	173
Dealing with Risk	173
Risk Management Life Cycle	174
Risk Identification	174
Ten Effective Methods to Identify Risks	174
Brainstorming	174
Survey	175
Interview	175
Practical Experience and Understanding	175
Research	176
Potential Risk Lists	176
Lessons Learned	176
Risk-Oriented Analysis	176
Design Template	176
Risk Assessment	176
What Is Risk Assessment?	177
Risk Assessment Process	177
Risk Assessment Involves Identified Risks	178
Technology Risk Assessment and Mitigation	
(TRAM) (Sample)	178
Business Risk	178
Catastrophic (A)	179
Critical (B)	179
Moderate (C)	179
Minor (D)	179
Risk Assessment Matrix	181
Negligible (E)	181
Risk Response	181
Avoid	181
Transfer	181
Reduce	182
Accept	182
Risk Mitigation	182
Risk Contingency Plan	183
Technology Contingency Plan (TCP) (Sample)	184

Application Risk Questionnaire (ARQ)	184
Project Risk Log	184
Part 2: Vulnerability, Risk, and Threat Analysis	184
Vulnerability and Risk	185
Step 1: Determine What Is Being Protected and Why	185
Sample Statement	186
Step 2: Identify the System	186
Step 3: Characterize System Operations	187
Step 4: Ascertain What One Does and Does Not Have Control Over	187
Vulnerability and Threat	188
Definitions	188
Four Levels of Threats	188
Four Steps of Risk Assessment	189
Step 1: Analysis Techniques Are Selected and Used	189
Step 2: Identify Vulnerabilities, Their Type, Source, and Severity	190
Step 3: Identify Threats, Their Type, Source, and Likelihood	190
Step 4: Evaluate Transaction Paths, Threat Zones, and Risk Exposure	190
Part 3: OCTAVE and Risk Management	192
What Is OCTAVE?	192
OCTAVE Phases	194
Phase 1: Build Asset-Based Threat Profiles	194
Phase 2: Identify Infrastructure Vulnerabilities	195
Phase 3: Develop Security Strategy and Plans	195
OCTAVE Way of Risk Management	195
OCTAVE in Risk Management	196
Appendix A—Sample	197
Vulnerability/Risk Assessment	197
For Pharmacy Handheld Technology	197
Introduction	197
Statement of Goals	197
High-Level System Entity Control Analysis	197
Vulnerability and Threat Analysis	200
Physical Structure	202
Virtual Private Network as a Risk	202
The Major Strengths of Utilizing Internet-Based VPN Services	203
Assumptions	203
Appendix B	204
Risk Factors Assumptions	204
Investment Size	204
Management Process Maturity	205

Degree of Technical Risk	206
Return Factors	206
Conclusion	206

## SECTION IV SOFTWARE QUALITY EXPECTATION

<b>CHAPTER 9 INFORMATION SECURITY</b>	211
Introduction	211
Part 1: Definition and Importance	211
What Is Information Security?	211
Difference between Privacy and Security	213
Key Points on Information Security	213
From What Threats Does Information Need to Be Secured?	213
Cybercrime	213
Types of Cybercrime	214
Computer Virus	214
Scam	215
Money Laundering	215
Phishing	215
What Kind of Information Needs to Be Secured	217
Some Examples of Recent Phishing	217
Identity Theft	219
Information That Is Considered Identity	220
Social Security Numbers	220
Date of Birth	220
Current and Previous Addresses and Phone Numbers	220
Current and Previous Employment Information	221
Financial Account Information	221
Mother's Maiden Name	221
Other Personal Information	221
Password for Nonfinancial Accounts	221
Password for Financial Accounts	221
Criminal Activities That Lead to Cybercrime	221
Spyware	221
Objective of Information Security	222
Why Is Security Important?	222
What Is the Benefit of Information Security?	224
Part 2: Methodology	224
The Strategy	224
Security Standards	224
ISO 15408	224
Control Objectives for Information and (Related) Technology (COBIT)	225
ISO 17799/BS7799	225
COBIT	225
OCTAVE	225

ISO 15408 vs. ISO 17799	225
Security Policy	225
Organizational Security	226
Asset Classification and Control	226
Personnel Security	227
Physical and Environmental Security	227
Communications and Operations Management	228
Access Control	229
System Development and Maintenance	229
Business Continuity Management	230
Compliance	230
Precautionary Guidelines	230
Refrain from Giving Out Personal Information	231
Storing Financial Records	231
Use Firewall Programs	231
Do Not Open Files Sent from an Unknown Source	231
Use a Secure Browser	231
Delete All Stored Personal Information	232
Do Not Disclose Passwords to Anyone	232
Beware of Phishing, Spoofing, and Spam Attempts	232
COBIT Security Baseline	232
Business Model Information Security	232
The Broader Scope of InfoSec	234
Operational Procedure for Doctor	234
Operational Procedure for Pharmacy	235
Common Information Security Criteria	236
Operational Procedure for Patient	237
Operation Procedure for Pharmacy Hub	237
Operational Change Control	237
Incident Management Procedure	238
External Facilities Management	239
System Planning and Acceptance	239
Capacity Planning	239
System Acceptance	239
Protection against Malicious Software	240
Control against Malicious Software	240
Housekeeping	240
Information Backup	240
Operator Logs	241
Fault Logging	241
Network Management	241
Network Controls	241
Media Handling and Security	242
Management of Removable Computer Media	242
Disposal of Media	242
Exchange of Information and Software	242
Security of Media in Transit	243

Electronic Commerce Security	243
Security of Electronic Mail	243
Business Requirement for Access Control	243
Access Control Policy	243
User Access Management	243
User Registration	243
Privilege Management	244
User Password Management	244
Review of User Access Rights	245
User Responsibilities	245
Network Access Control	246
Policy on Use of Network Services	246
Remote Diagnostic Port Protection	246
Network Connection Control	247
Operating System Access Control	248
Automatic Terminal Identification	248
Terminal Log-On Procedures	248
User Identification and Authentication	248
Password Management System	248
Use of System Utilities	248
Duress Alarm to Safeguard Users	248
Terminal Time-Out	249
Limitation of Connection Time	249
Application Access Control	249
Information Access Restriction	249
Sensitive System Isolation	250
Monitoring System Access and Use	250
Event Logging	250
Monitoring System Use	250
Clock Synchronization	250
Mobile Computing and Teleworking	250
Mobile Computing	250
Teleworking	251
Security Requirements of Systems	251
Security in Application Systems	251
Data Validation	251
Business Continuity Management	252
Aspects of Business Continuity Management	252
Primary Focus of the Plan	252
Primary Objectives of the Plan	253
Plan	253
Personnel	254
Salvage Operations at the Disaster Site	254
Designate Recovery Site	254
Purchase New Equipment	254
Begin Reassembly at the Recovery Site	255
Restore Data from Backups	255



Restore Applications Data	255
Move Back to Restored Permanent Facility	255
Compliance	256
Compliance with Legal Requirements	256
Identification of Applicable Legislation	256
Intellectual Property Rights	256
Copyright	256
Reviews of Security Policy and Technical Compliance	258
System Audit Considerations	259
System Audit Controls	259
Protection of System Audit Tools	259
Part 3: Security Policy Document	259
Information Security Policy	260
Board-Level Action	261
Management-Level Action	262
Organizational Security	262
Information Security Infrastructure	262
Management Information Security Forum	263
Information Security Coordination	263
Allocation of Information Security Responsibilities	263
Authorization Process for Information Processing	
Facilities	263
Specialist Information Security Advice	264
Cooperation between Organizations	264
Independent Review of Information Security	264
Security of Third-Party Access	264
Identification of Risks from Third-Party Access	264
Types of Access	264
Reasons for Access	265
On-Site Contractors	265
Security Requirements in Third-Party Contracts	265
Outsourcing	265
Security Requirements in Outsourcing Contracts	265
Asset Classification and Control	265
Accountability for Assets	265
Inventory of Assets	266
Information Classification	266
Classification Guidelines	266
Information Labeling and Handling	266
Personnel Security	268
Security in Job Definition	268
Personnel Screening Policy	268
Testing Employees	268
Evaluate Key Job Behaviors	268
Confidentiality Agreements	269
Terms and Conditions for Employment	269
User Training	270

Information Security Education and Training	270
Reporting Security Incidents	270
Security Incidents Reporting Guideline	270
Reporting Security Weaknesses	270
Physical and Environmental Security	271
Physical Security	271
Physical Entry Control	271
Securing Offices, Rooms, and Facilities	271
Equipment Security	272
Protect the System from Undesirable Booting	272
Set Up Storage Protection for Backup Tapes	273
Equipment Sitting and Protection	273
Power Supplies	273
Cabling Security	273
Equipment Maintenance	273
General Controls	273
Clear Desk and Clear Screen Policy	273
Removal of Property	274
Communication and Operation Management	274
Operational Procedure and Responsibilities	274
Documented Operating Procedures	274
Information Security Certification Procedure (Sample)	274
Document Change Control Log	275
Security Standards	276
ISO 15408	276
COBIT	276
ISO 17799/BS7799	276
OCTAVE	276
<b>CHAPTER 10 INFORMATION AUDIT</b>	277
Introduction	277
Part 1: Definition and Planning	277
Definition	277
Audit Planning	279
IT Audit Plan Development Process	281
Role of Supporting Technologies	281
Understanding the Business	282
Operating Environment	282
Details of the IT Audit	283
Examining the Business Model	283
Formalizing the IT Audit Plan	283
Integration of the IT Audit Plan	284
Validating the Audit Plan	284
The IT Audit Plan Should Be Dynamic	284
Ten Key IT Considerations for Internal Audit	284
Responsibilities of IT Audit Team Members	285
Lead Auditor	285
Recorder	286

Auditor	286
Initiator	286
Audited Organization	286
Auditor's Qualifications	286
Choosing an Auditor	286
Auditor's Education	287
Knowledge and Skills	287
Experience	288
Knowledge	288
Talent	288
Competence	288
Part 2: Audit Process and Procedure	288
Audit Process	289
Audit Process Implementation	290
Support for the Audit Process	290
Procedures	290
Management Preparation	290
Verification of Quality Manual	291
Verification of Implementation of the	
Quality Manual	291
Sample Work Instructions	293
Postimplementation Review	293
Key Phase Review	294
Project Management Methodology Assessment	294
Privacy and Audit Management	295
Five Key Focus Areas for Project Audits	295
Business and IT Alignment	296
Project Management	296
IT Solution Readiness	297
Solution Design	297
Organizational and Process Change Management	297
The Audit Report	298
Part 3: Auditing and Information Security	299
Defined and Planned Strategy	299
Auditing Privacy Risks	299
Auditing Data Categorization	300
Auditing Law and Regulation Aspects	301
Organization Threats	301
Application Risks	301
Business Process Risks	302
Auditing IT Vulnerabilities	302
Identifying Insignificant Vulnerability Management	302
The Internal Auditor's Role About	
Information Security	303
Vulnerability and Risk	303
Persistent Auditing and Monitoring	304
Suggested Readings	305

**CHAPTER 11 SOFTWARE RELIABILITY AND PROCESS**

<b>IMPROVEMENT</b>	307
Introduction	307
Part 1: Definition and Measurement	307
What Is Reliability?	307
What Are Reliability Metrics?	307
Classifications	307
Standards Defining Reliability Measurement	308
Selection of Measures	308
Measures from IEEE 982.2	308
Measurement-Based Assurance	309
Criteria for Selection	309
Sample Primitive Metrics	309
Primitive Cost and Effort Metrics	310
Primitive Change Metrics	310
Software Requirements Metrics	310
Requirements Size Metrics	310
Requirements Traceability	310
Completeness	311
Fault-Days Number	311
Software Design Metrics	311
Primitive Size Metrics	312
Primitive Fault Metrics	312
Primitive Complexity Metrics	312
Defect Density	312
Test-Related Primitives	313
Code Metrics	313
Cyclomatic Complexity (C)	313
Amount of Data	314
Live Variables	314
Test Metrics	314
Fault Density	314
Defect Age	315
Defect Response Time	315
Defect Cost	315
Defect Removal Efficiency	315
Primitive Test Case Metrics	316
Statement Coverage	316
Branch Coverage	316
Path Coverage	316
Data Flow Coverage	316
Test Coverage	316
Mean Time to Failure	317
Failure Rate	317
Cumulative Failure Profile	317
Customer Ratings	318
Customer Service Metrics	318

Making Reliability Metrics Meaningful	318
Standards Defining Software Measurement	318
Productivity Metrics: IEEE 1045	319
Software Reliability: IEEE 982	319
Quality Metrics Methodology	320
IEEE 1061–1992	320
Software Reliability Measurement	321
What Is a Model?	321
Qualities of a Good Model	321
The Importance of Data	321
Metrics and Models	322
Model Development and Independent Metrics	322
The Issue of Availability	322
Data Retention and Use	322
Validity	323
Software Reliability Estimation	323
CMMs: The Software Engineering Institute's	
Capability Maturity Model	323
Maturity Levels	323
Initial	324
Repeatable	324
Defined	324
Managed	324
Optimized	324
Common Features	325
CMMI	325
Staged Representation	325
Continuous Representation	325
Disciplines and Environments	326
CMMI Application	326
Maturity Levels	326
Process Areas	326
Level Three Process Areas	327
Level Four Process Areas	327
Level Five Process Areas	327
IDEAL	327
Part 2: Software Process Improvement and Capability	
Determination (SPICE)	328
ISO 15504 and Management	328
The Assessment Process	328
The Reference Model	328
The Capability Dimension	329
The Engineering Process Category	329
The Project Process Category	330
The Support Process Category	330
The Organization Process Category	330
ISO/IEC 15288 Processes	330

ISO 15288 Relation to Other Frameworks	331
Personal and Team Approaches	332
PSP and TSP to CMM	332
The PSP Process Structure	333
PSP Quality Management	333
Early Defect Removal	333
Defect Prevention	334
PSP Project Plan Summary	334
Outcomes of the Process	335
The Team Software Process	335
Definition	335
The TSP Team Working Process	336
What Does TSP Do for Software?	337
Measurement	337
Application	338
TSP Quality Management	338
The Quality Plan	338
Identifying Quality Problems	339
Finding and Preventing Quality Problems	339
Relationship of PSP and TSP to CMM	339
Appendix	340
Software Process Improvement	340
Introduction	340
Purpose	340
Scope	340
Assumptions	341
Constraints	341
Compliance	341
Acronyms and References	341
Acronyms	341
Organization and References	342
INDEX	343

# DEFECT MANAGEMENT

## Introduction

This chapter, as the name implies, deals with the conceptual aspects of defect management. There are three parts in this chapter. Part 1 discusses the basic concepts of a defect and why a defect happens. Part 2 introduces the practical methodologies of how to manage the defects. In this section, some sample documents and templates are provided to manage the defect properly. Part 3 discusses and analyzes the root causes of defects and provides recommendations of how to prevent defects in the future.

## Part 1: Definition and Analysis

### *Definitions*

*Defect* A defect in simple terms is a variance from expectation. Another definition is that a defect is a condition in a process/product which does not meet a documented requirement. In other words, a defect is an error in a process or product's behavior that causes it to malfunction or to produce incorrect or unexpected results.

The root cause of a defect may originate from different sources such as code, requirements, design, environment, build/compilation, test case, and data.

*Defect in Hardware* In IEEE 610, defect or fault is defined as "A defect in a hardware device or component; for example, a short circuit or broken wire."

*Defect in Software* “An incorrect step, process, or data definition in a computer program.”\*

This definition is used primarily by the fault tolerance discipline. In common usage, the terms “error” and “bug” are used to express this meaning.

#### *Definition of an Error*

In IEEE 610, the error is defined as

- “The difference between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition.”
  - For example, a difference of 30 m between a computed result and the correct result.
- “An incorrect step, process, or data definition.”
  - For example, an incorrect instruction in a computer program.
- “An incorrect result.”
  - For example, a computed result of 12 when the correct result is 10.
- “A human action that produces an incorrect result.”
  - For example, an incorrect action on the part of a programmer or operator.†

*Defect Repository* Defect repository is the defect management tool/repository used to track defects and all defects associated with the application under test. There are many tools available. However, many companies typically use HP Quality Center or IBM Clear Quest for defect repository.

#### *What Causes Defects in Software*

Certainly, it is very important and necessary to understand why an error happens. Honestly speaking, that is the point. When the cause

---

\* IEEE Std 610.12-1990 (Revision and redesignation of IEEE Std 75Z.1983) *IEEE Standard Glossary of Software Engineering Terminology*, p. 32.

† Ibid., p. 31.



could be identified, almost half of the problem is resolved. In this arena, it is called defect root cause. In this chapter, we have dedicated one important part on root cause analysis (RCA), and we will try to evaluate them.

When the software code has been built, it is executed and then any defects may cause the system to fail to do what it should do (or do something it should not), causing a failure. Interestingly and alarmingly, sometimes a defect may not be obvious even though it exists; in programming language, it is called a logic error.

In computer programming, a logic error is a bug in a program that causes it to operate incorrectly, but not to terminate abnormally (or crash). A logic error produces unintended or undesired output or other behavior, although it may not immediately be recognized as such.

Logic errors may occur in both compiled and interpreted languages. Unlike a program with a syntax error, a program with a logic error is a valid program in the language, even though it does not behave as intended. The only clue to the existence of logic errors is the production of wrong solutions.

This example function in C is to calculate the average of two numbers that contains a logic error. It is missing brackets in the calculation so it compiles and runs but does not give the right answer due to operator precedence (division is evaluated before addition).

```
int average (int a, int b)
{
    return a + b / 2;    /* should be (a + b) / 2*/
}
```

In simple math,

$(2 + 5) \times 2$  and  $2 + 5 \times 2$  is not the same.

In  $(2 + 5) \times 2$  the result is 14; on the other hand,  $2 + 5 \times 2 = 12$  because here you have to do the multiplication first then the addition. There is a rule called PEMDAS which represents as

P = Parenthesis  
E = Exponents  
M = Multiplication  
D = Division  
A = Addition  
S = Subtraction

In these circumstances, if the developer writes the wrong code such as forgetting to put  $(2 + 5)$  in parenthesis, then the result will be wrong even though for the tester it may look like the correct result.

### *Detecting a Defect Early*

It is indeed better to find the defect as early as possible. In software development, if there is a mistake in requirement and you found it in production that could lead to a big mess.

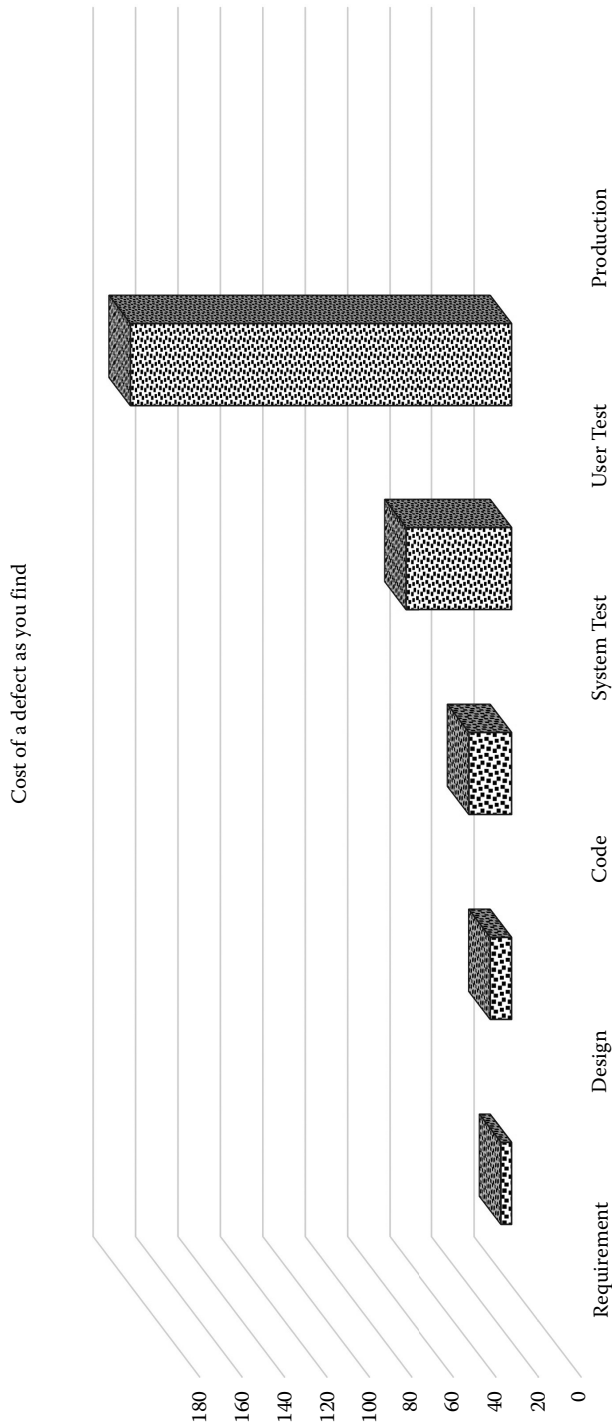
To fix this defect, no matter how much it may cost, you may not find all people you need—developer, tester, designer—everyone may not be available then, and it is not a simple thing. For example, it is easier to build a new building than repair an old building. It could be a total disaster.

### *What Is the Cost of Defects Not Detected Early?*

In addition, considering the impact of failures arising from defects, which we have not found, we need to consider the impact once we find those defects. The cost of finding and fixing defects rises considerably across the life cycle; think of the old English proverb “a stitch in time saves nine.” This means that if you mend a tear in your sleeve now while it is small, it is easy to mend; but if you leave it, it will get worse and need more stitches to mend it.

When a defect exists in the requirement specification and is not detected until acceptance testing or until production, then it will be much more expensive to fix (Figure 7.1).

It is quite often the case that defects detected at a very late stage are not corrected because the cost of doing so is too expensive. Also, if the software is delivered and meets an agreed specification, if the specification was wrong, then the software will not be accepted. The project team may have delivered exactly what they were asked to deliver, but it is not what the users wanted. In some cases, where the defect is too serious, the system may have to be completely reinstalled.



**Figure 7.1** The cost of a defect is much higher in production than in requirement.

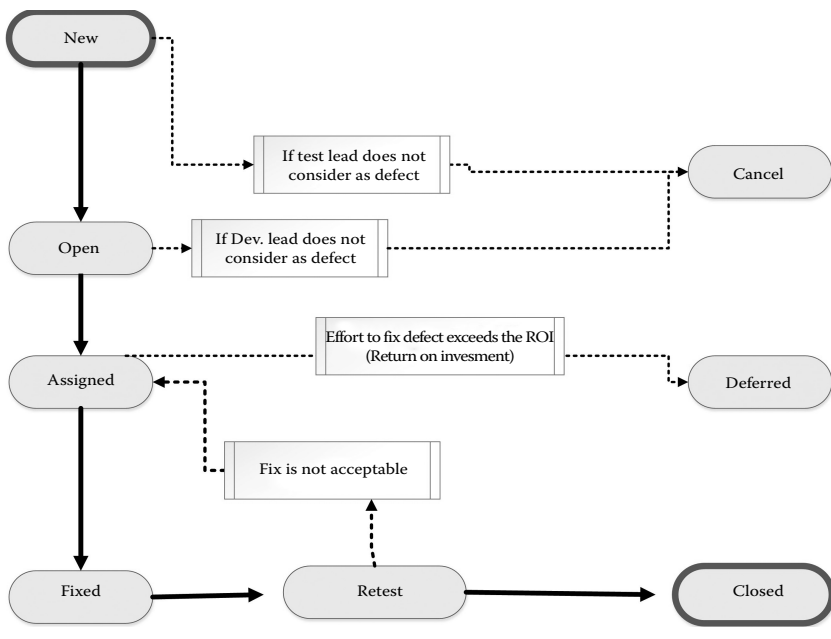
*Defect Life Cycle Steps*

The IEEE 1044 defect life cycle consists of the following four steps (Figure 7.2):

*Step 1: Recognition or Identification* Recognition occurs when we observe an anomaly, that observation being an incident, which is a potential defect. This can occur in any phase of the software life cycle.

*Step 2: Investigation* After recognition, the investigation of the incident occurs. Investigation can reveal related issues. Investigation can propose solutions. One solution is to conclude that the incident does not arise from an actual defect; for example, it might be a problem in the test data.

*Step 3: Action* The results of the investigation trigger the action step. We might decide to resolve the defect. We might want to take action indicated to prevent future similar defects. If the defect is resolved,



**Figure 7.2** Defect life cycle from new to closed.

regression testing and confirmation testing must occur. Any tests that were blocked by the defect can now progress.

*Step 4: Disposition* With action concluded, the incident moves to the disposition step. Here, we are principally interested in capturing further information and moving the incident into a terminal state.

### *Objectives of Testing*

*Reduce the Risk of Failure* Most of the complex software systems contain faults, which cause the system to fail from time to time. This concept of “failing from time to time” gives rise to the notion of *failure rate*. As faults are discovered and fixed while performing more and more tests, the failure rate of a system generally decreases. Thus, a higher level objective of performing tests is to bring down the risk of failing to an acceptable level.

*Reduce the Cost of Testing* The different types of costs associated with a test process include the cost of designing, maintaining, and executing test cases; the cost of analyzing the result of executing each test case; the cost of documenting the test cases; and the cost of actually executing the system and documenting it.

Therefore, the fewer test cases designed, then the cost of testing is reduced. However, producing a small number of arbitrary test cases is not a good way of saving money. The highest level objective of performing tests is to produce low-risk software with fewer test cases. This idea leads us to the concept of *effectiveness of test cases*. Therefore, the test engineers must judiciously select fewer, more effective test cases.

### *Analyze Root Causes*

According to Capability Maturity Model Integration (CMMI), the objective of defect RCA is to determine causes of defects.

Root causes of defects and other problems are systematically determined.

*Address Causes of Defects* Root causes of defects and other problems are systematically addressed to prevent their future occurrence.

*Institutionalize a Defined Process* A root cause is a source of a defect; if it is removed, the defect is decreased or removed.

Determine which defects and other problems will be analyzed further.

When determining which defects to analyze further, consider the impact of the defects, the frequency of occurrence, the similarity between defects, the cost of analysis, the time and resources needed, safety considerations, and so on.

Perform causal analysis of selected defects and other problems and propose actions to address them.

The purpose of RCA is to develop solutions to the identified problems by analyzing the relevant data and producing action proposals for implementation.

Conduct causal analysis with the people who are responsible for performing the task.

Causal analysis is performed with those people who have an understanding of the selected defect or problem under study, typically in meetings.

An action proposal usually documents the following:

- Originator of the action proposal

- Description of the problem

- Description of the defect cause

- Defect cause category

- Phase when the problem was introduced

- Phase when the defect was identified

- Description of the action proposal

- Action proposal category

Projects operating according to a well-defined process will systematically analyze the operation where problems still occur and implement process changes to eliminate root causes of selected problems.

*Implement the Action Proposals*

Implement the selected action proposals that were developed in causal analysis.

Action proposals describe the tasks necessary to remove the root causes of the analyzed defects or problems and avoid their recurrence.

Only changes that prove to be of value should be considered for broad implementation.

**Part 2: Process and Methodology***Defect Management Process*

There are several high-level steps to be taken in a typical defect management process. The following items are highly recommended, which are also supported by IEEE standards.

*Identifying*

The first thing that needs to be done is to identify the defect: what is it and how did this happen? The first person who identifies the defect should submit it as defect to his or her lead and the team lead should evaluate, verify, and identify it as a defect, and then it remains an open defect.

*Categorizing*

When a defect is reported and verified by the test team, it remains as open, then it should be assigned to someone, usually to a related developer. Once the defect is categorized, the defect moves on in the process to the next step that is prioritization.

*Prioritizing*

Prioritization is typically based on a combination of the severity of impact on the user, relative effort to fix, along with a comparison against other open defects. The priority should be determined

with representation from the management, the customer, and the project team.

### *Assigning*

Once a defect has been prioritized, it is then assigned to a developer or other technician to fix.

### *Resolving*

The developer fixes (resolves) the defect and follows the organization's process to move the fix to the environment where the defect was originally identified.

### *Verifying*

Depending on the environment where the defect was found and the fix was applied, the software testing team or customer typically verifies that the fix has actually resolved the defect.

### *Closing*

Once a defect has been resolved and verified, the defect is marked as closed.

### *Management Reporting*

Management reports are provided to appropriate individuals at regular intervals as defined reporting requirements. In addition, on-demand reports are provided on an as-needed basis.

## **Roles and Responsibilities in Software Development Life Cycle**

### *Business Owner*

The business owner requests funding, sets business requirements, and works with the technology owner to make strategic decisions.

### *Stakeholders*

Stakeholders include anyone who will be impacted by a project, including security, risk, compliance, and governance organizations.



Stakeholders should actively work with the analysts, testers, and developers to ensure that the defects have been logged and addressed in a timely manner, participate in defect review meetings, and provide input into the final defect disposition.

### *Analyst*

The analyst's role is responsible for reviewing any defects that impact business and system operations. The analyst should participate or represent someone in defect review meetings to ensure that proper severity and priority have been assigned to the defects. The analyst should work with the testing and development team to confirm that the defects have been properly fixed and retested.

### *Developer*

A developer is responsible for researching and remediating any assigned defects that have been opened by the testers or any other stakeholders. Developers should work with the testers and analysts to provide additional defect information, research the defect, and provide a fix to prevent the defect from recurring. Developers must participate in defect review meetings and provide updates to the defect fixes that are pending disposition as well as discuss any temporary workarounds that apply until a permanent fix is identified and implemented.

### *Tester*

A tester is a project team member performing testing activities such as system testing or user testing. Testers are responsible for testing the application, registering and tracking all testing defects, and documenting any issues that will need to be escalated or reviewed with the management. Testers should work with the business and development team to determine priorities, severities, and remediation dates. Testers must participate in defect review meetings to ensure that all defects are tracked and are appropriate.

### *Conflict Resolution and Escalations during Defect*

If there is any dispute or disagreement regarding any defect or about the interpretation of any terminology, and if the dispute cannot be

resolved within the business unit, the business owner shall attempt to resolve the dispute.

Defect Management Methodology

Identifier:  
Effective Date: mm/dd/yyyy  
Version: 0.00

Document Change Control

VERSION CHANGE DATE	VERSION #	WHAT KIND OF CHANGE/ REVISION	WHERE CHANGED HAPPENED (SECTION/PAGE)	REVISED BY NAME AND TITLE	APPROVED BY NAME AND TITLE

Documentation

PROCEDURE NAME	DEFECT MANAGEMENT PROCEDURE	COMMENT
Version number:	1.00	
Procedure identifier:		
Superseded procedure(s):	N/A	
Date approved:		
Effective date:		
Procedure author(s):		
Procedure owner:		
Procedure approver:		
Procedure repository:		
Supporting documentation:	Defect management standard Software development life cycle (SDLC) Standard End user computing (EUC) standard Security vulnerability remediation standard Infrastructure hardware change (IHC) procedure Incident management standard Incident management procedure Technology incident management standard Technology incident management procedure	

*Statement of Purpose*

The primary goal of this procedure is to provide clear definitions and a list of values for all software defect attributes. This procedure will ensure that all defect repositories will use a consistent defect reporting and management process.

*Risks*

It is important that the company or team exhibit suitable and effective controls managing defects, ensuring their timely resolution based on their severity, and update their resolution progress to the stakeholders as it is critical for the company's finance and other key business processes.

*Defect Steps*

STEP #	STATUS	DESCRIPTION	PRIMARY PERFORMER	OUTPUT/ EVIDENCE	INPUT
<i>1. SUBMITTING THE DEFECT</i>					
1	New	The submitter identifies and records in a defect repository.	Test team Business team	Defect record is "submitted" in the defect repository.	
<i>2. RESOLVING THE DEFECT (PENDING)</i>					
2	Open pending resolution	Acknowledge the submitted defect. The defect is assigned to the development team.	Development team/ business team	Defect record is moved to a "pending resolution" state in the defect repository.	
<i>3. RESOLVING THE DEFECT (RESOLVED)</i>					
3	Fix and resolved	Resolve the acknowledged defect. The defect is sent to the submitter for re-test.	Development team	Defect record is moved to "resolved" state in the defect repository.	

*(Continued)*

STEP #	STATUS	DESCRIPTION	PRIMARY PERFORMER	OUTPUT/ EVIDENCE	INPUT
4. REOPENING THE DEFECT					
4	Reopen pending resolution	The submitter retests the resolved defect and the defect still exists. Reopen the resolved defect. The defect is reassigned to the development team for further analysis or resolution.	Test team Business team	Defect record is moved to a “pending resolution” state in the defect repository.	
5. CLOSING THE DEFECT					
5	Closed	The submitter retests the resolved defect and the defect does not exist. Close the resolved defect.	Test team Business team.	Defect record is moved to “closed” state in the defect repository.	
6. DEFERRING THE DEFECT					
6	Deferred	Business team determined to defer the acknowledged defect.	Business team	Defect record is moved to “deferred” state in the defect repository.	
7. CANCELING THE DEFECT					
7	Cancelled	Business or development team cancels the acknowledged or deferred defect. Submitter concurs.	Test team Business team	Defect record is moved to “cancelled” state in the defect repository.	

Defect States

The mandatory states of a defect are

- Submitted/New
- Open
- Resolved
- Closed
- Canceled
- Deferred

			OPTIONAL (O) REQUIRED (R) CONDITIONALLY REQUIRED (CR)	BUSINESS RULES
ATTRIBUTES	DESCRIPTION	TYPE		
Defect ID	Defect name of ID	System list	R	
Project ID	Project name or ID	System list	R	Name or ID of the project. Most recent project name or ID must be used if a defect is found in production.
Application	System in which defect was identified	List	R	Unique application identifier. Should match to the CMDB and asset ID databases.
Functional area	Module/ subsystem/ component in which defect was identified	System list	R	List of functional areas, modules, or components. The defect is associated with an application.
Headline	One line summary of the defect	Text	R	
Description	Detailed description of problem that includes steps to reproduce the defect, actual results, and expected results	Text	R	
SDLC phase found	Phase where the defect was detected	List	R	Analysis (optional), design (optional), construction, system test, user test, implementation and postimplementation.
Found in environment	Environment in which defect was found	List	CR	Development, test, acceptance, production, integration and contingency. Not required if root cause is requirement or design.
Found in release number	Release number in which the defect is found	System list	R	

*(Continued)*

ATTRIBUTES	DESCRIPTION	TYPE	OPTIONAL (O) REQUIRED (R) CONDITIONALLY REQUIRED (CR)	BUSINESS RULES
Closed in release number	Release number for which the defect is closed	System list	CR	Conditionally required if the state is closed
Remedy incident ID	Problem ticket number for defects found in production	Text	CR	Conditionally required if the SDLC phase found is implementation or postimplementation
Test case ID	Associated test case ID	Text	CR	Not required if root cause is requirement or design
Functionality type	Lists the type of functionality that introduced the defect	List	R	New functionality, existing functionality
Severity	Impacts to application functionalities, business processes, or interfaces causing minor to critical disruption to application usage	List	R	Severity of the defect must be set in consensus with stakeholders to one of the following: Critical High Medium Low

*Defect Attributes*

When a defect is discovered, the following minimum set of defect information must be reported in the defect repository.

ATTRIBUTES	DESCRIPTION	TYPE	OPTIONAL (O) REQUIRED (R) CONDITIONALLY REQUIRED (CR)	BUSINESS RULES
Priority	Prioritizing defect based on how fast the defect should be fixed	List	R	Priorities will be set by the submitter of the defect to one of the following: Critical High Medium Low

(Continued)

ATTRIBUTES	DESCRIPTION	TYPE	OPTIONAL (O) REQUIRED (R) CONDITIONALLY REQUIRED (CR)	BUSINESS RULES
Root cause	Analysis of what caused the defect	List	R	
Root cause reason	Provide the reason for the root cause of the issue.	List	CR	Required if root cause is: Requirements Incomplete/missing Unclear Inconsistent Incorrect Not traceable Not testable
Resolution notes	Details regarding the resolution of defect	Text	CR	Conditionally required if the state is not submitted
State	Provides a current state of the defect's current flow while going through defect resolution process	System action	R	This is the defect state at any point in time. The values in this field are auto populated
Deferral Business Impact	Business impact description if defect is deferred as well as work around if applicable	Text	CR	Required if the defect is deferred
Submitter	Name of person that created the defect	System generated	R	
State updated By	Name of person that last changed the state of the defect	System list or system generated	CR	Name of the person that last change the state. Required if the state of the defect is deferred, closed, or canceled.
Update date	Date on which defect state	System	R	Defect repository will maintain the audit trail for the defect
Workaround	Lists the work around for deferred defects	Text	CR	Required if the state of the defect is deferred
System test	Identifies whether the defect was leaked from system test	List	CR	Required if the SDLC phase found is user test of higher  Yes No

Defect Priorities

The defect priorities and definition may differ from one testing stage to another; sometimes in some projects, it may also differ from person to person. The basic definitions are provided below. The expected resolution timeframe for the defects depends on their priority.

Security defects have a prescribed timeframe for remediation that are spelled out in the security, vulnerability, and remediation standard.

PRIORITY DESCRIPTION	PRIORITY DEFINITIONS FOR DEFECTS IN NONPRODUCTION	PRIORITY DEFINITIONS FOR PRODUCTION DEFECTS
Critical	Immediate attention—critical may also mean that it might be blocking some other activities. The critical defect should be resolved immediately.	Immediate attention—must receive highest development priority and should be resolved immediately.
High	Should be reported immediately to the development team. A response or action plan must be provided within 2 working days since the defect causes more than one of the functional areas to be untestable.	Should be reported immediately to the development team. A response or action plan must be provided within 2 working days.
Medium	A response or action plan should be reported within 5 working days.	A response or action plan should be reported within 5 working days. This defect should be resolved in the next release.
Low	Fix dates are subject to negotiation. An action plan before the next release.	Fix dates are subject to negotiation. An action plan before the next release.

Defect Severities

The defect severity definition may also differ from one testing stage to another and also may differ among stakeholders as human perception may be different.

SEVERITY DESCRIPTION	SEVERITY DEFINITIONS FOR SYSTEM, USER, AND PRODUCTION	SEVERITY DEFINITIONS FOR REQUIREMENT DEFECTS
Critical	Critically severe defect causes severe business disruption, financial or reputational impact, and no workaround exists. The customer is unable to use the product, resulting in a critical impact to their operation. This defect must be resolved before exiting current phase or releasing to production.	The reason for the requirement defect could be considered such as Incomplete/missing Inconsistent Incorrect

(Continued)



SEVERITY DESCRIPTION	SEVERITY DEFINITIONS FOR SYSTEM, USER, AND PRODUCTION	SEVERITY DEFINITIONS FOR REQUIREMENT DEFECTS
High	Significant business disruption but a workaround exists. The customer is able to use the product but is severely restricted. This defect should be resolved before exiting current phase or releasing to production.	Content has a major inaccuracy or is missing important detail. The reason for the requirement defect could be considered such as Incomplete/missing Incorrect Unclear Inconsistent Not traceable Not testable
Medium	Minor business disruption but has a workaround, minor usability issues. This defect should be resolved before exiting current phase or releasing to production.	Content is correct but has a moderate flaw that needs amendment; for instance, because it is unclear, imprecise, or not concise. The reason for the requirement defect could be considered such as Unclear Not traceable Not testable
Low	The defect may be cosmetic in nature or a usability annoyance, such as warning messages, misspelled words, etc.	Formatting or organizational observation or a grammatical or spelling error not affecting the meaning. The reason for the requirement defect is usually unclear

### Part 3: Root Cause Analysis

#### *Definition*

A root cause is an originating cause of either a condition or a causal chain that leads to an outcome or result.

In software development, we can see how defects may arise and are caused by any field.

Root cause of a defect identifies the process or source that introduced the defect.

#### *Root Cause Fields*

Standard acceptable values for the root cause field are listed in the following sections:

*Requirements* This field is required if the root cause of the defect is requirements and should list the actual issue of the stakeholder requirement document that introduced the defect.

*Defect Cause in Requirement* The possible cause of defect in requirement could be

*Incomplete/Missing* Necessary functionality is omitted from the requirements set. The defect should specify what needs to be documented to address the gap.

Unclear: A requirement is not simple, specific, clear, and unambiguous.

*Inconsistent* A requirement is in conflict with one or more other requirements or other requirements make it redundant.

*Incorrect* A requirement does not reflect the needs of one or more project stakeholders.

*Not Traceable* A requirement cannot be traced to project scope, that is, it cannot be established as being within the approved scope of the project.

*Not Testable* A requirement does not specify observable functionality and so cannot be validated by testing.

*Implementation Dependent* A requirement does not describe desired system functionality independent of the technology and design that will be used to achieve it.

*Design* This root cause should be selected if the solution specifications or detailed design are missing, inconsistent with requirements, or otherwise incorrect.

*Code* This root cause should be selected if the application failed to produce expected result or the functionality is missing, not consistent

with stakeholder requirements, solution specifications, standards, or otherwise incorrect.

*Environment* This root cause should be selected if the application failed to produce expected results due to incorrect environment, infrastructure, or application configuration set up. Examples of environment issues are errors in software compilation/build, incorrect application configuration settings, application processes not initialized, application password is expired, third-party packages are missing, dependent systems are not available, and so on.

*Test* This root cause should be selected if a defect was reported incorrectly because of inconsistent test case results with stakeholder requirements or solution specifications, premature test case execution, or the test case was not executed in the appropriate environment.

*Data* This root cause should be selected if the product failed to produce expected results due to the improper setup of test data in the pertinent databases or input files.

*Analysis* RCA is an effective technique to investigate the origin for defect occurrence. This analysis helps to prevent reoccurrences of defect in future.

#### *The Most Common Root Cause Classifications*

Despite the existence of various rationales, RCA techniques enable to classify the most common root causes and percentage of their contributions toward various defect patterns. They are communication (25%–30%), education (20%–25%), oversight (30%–40%), transcription (20%–25%), and miscellaneous (5%–10%). From the defect distribution and defect pattern analysis, it is evident that trivial defects contribute more toward defect injection (Table 7.1).

**TABLE 7.1** A Sample Report of Defects in Different Stages and Severity Level Report, and % of Pass and Fail

PLANNING PHASE	
METRICS DATA ELEMENT	VALUE
Planned progression coverage	100%
Prerelease overall regression capability	90%
Prerelease overall regression automation	70%
Planned regression coverage	100%
Planned regression coverage—automated	40%
Expected initial pass rate	90%
EXECUTION PHASE	
METRICS DATA ELEMENT	VALUE
Actual progression coverage	100%
Progression coverage—automated	50%
Actual regression coverage	80%
Postrelease overall regression capability	
Postrelease overall regression automation	
Actual initial pass rate	80%
Final pass rate	100%
System test defects—critical/high	5
System test defects—medium/low	12
System test defects—deferred	2
PRE-PROD PHASE	
METRICS DATA ELEMENT	VALUE
UAT defects—critical/high	2
UAT defects—medium/low	3
UAT defects—deferred	1
System test leakage—critical/high	0
System test leakage—medium/low	1

In Table 7.1, in the planning phase:

- Progression coverage was planned to cover 100% and actually 100% was covered.
- Prerelease overall progression was planned to automate 70%, but the actual progression automation was covered 50%.
- Regression coverage was planned to cover 60%, but actually team was able to cover 80%.

Initial pass rate was expected (as usually some tests fail) to reach 90% (line 9); however, the actual initial pass rate was 80% (line 19),

which is less than expected. This also means more test cases failed than expected, but the good news is that the final pass rate is 100%, which means the development team was able to resolve and fix the defects.

There are 11 defects found by the system test team, where the severity level of 3 of them were critical/high (line 21) and 8 of them were medium/low (line 22); on the other hand, UAT or the user acceptance test team, found 5 defects in total, 4 which were already found by the test team, 1 defect that was found by UAT but the system test failed to find it defined and system test leakage.

So altogether, there were 12 defects; among these, 9 defects were resolved and closed, and 3 could not be resolved at this time. The team including stakeholders decided to work around for now and possibly resolve it in future release. These 3 unresolved defects are called deferred (Figures 7.3 and 7.4).

DEFECT ID	STATE	SDLC PHASE		DEFECT PHASE AGE
		FOUR	ROOT CAUSE	
1001	Closed	System Test	Requirement	3
1002	Closed	Design	Code	−1
1003	Closed	System Test	Design	2
1004	Closed	System Test	Requirement	3
1005	Closed	System Test	Code	1
1006	Deferred	System Test	Code	1
1007	Deferred	User Test	Code	2
1008	Closed	Design	Code	−1
1009	Closed	Design	Requirement	1
1010	Closed	System Test	Code	1
1011	Closed	User Test	Design	3
1012	Deferred	User Test	Code	4

### *Defect Prevention*

Awareness of defect injecting methods and processes enables defect prevention (DP). It is the most significant activity in software development. It identifies defects along with their root causes and prevents their recurrences in the future.

*Benefits of Defect Prevention* Prevention is better than a cure; it applies to defects as well. It is indeed better to prevent a defect

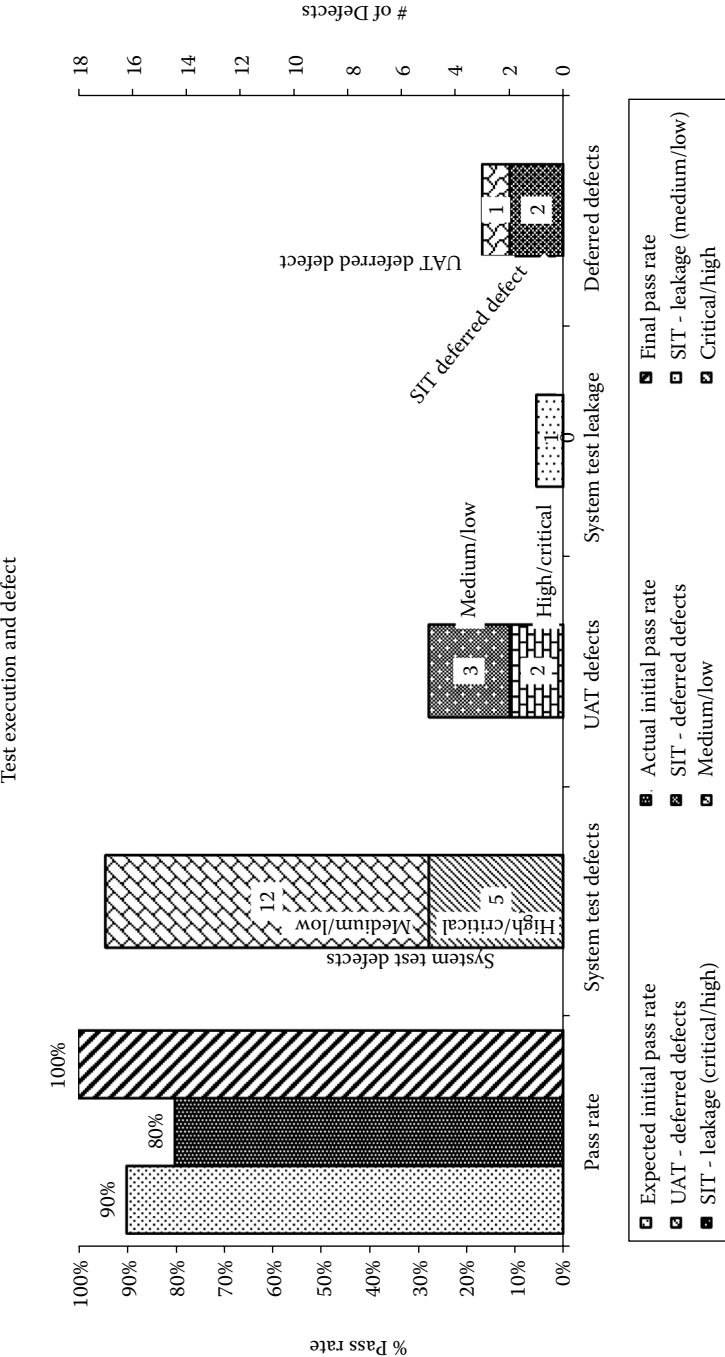


Figure 7.3 A sample report % of pass and defects.

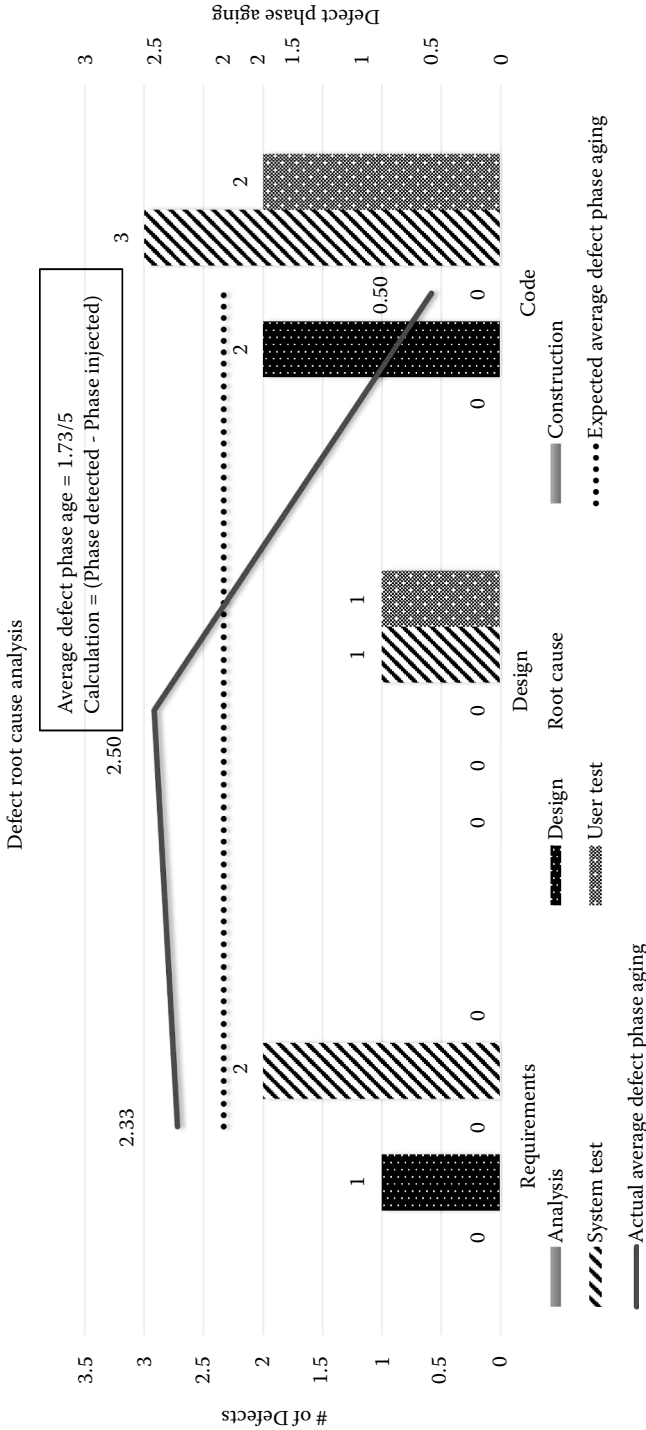


Figure 7.4 Defect root cause analysis, average phase age.

before it reaches its severity level. Validation, verification, inspection, and review helps to prevent defects or severe risk issues.

Therefore, it is imperative to introduce defect prevention at every level of SDLC to prevent defects at the earliest occurrence.

### *Defect Prediction*

Defect prediction is a technique of identifying the quality of the software before deployment. It improves the performance. The main objective is Software Quality Assurance.