## Contents

**Preface**  
xxv

**Content Overview**  
xxvii

**Acknowledgments**  
xxxii

**Author**  
xxxiii

### Section I  Concept

**Chapter 1  Quality Concept and Perspectives**  
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

Software Quality Assurance: Integrating Testing, Security, and Audit
https://www.crcpress.com/9781498735537
# Contents

- **Input**
  - When to Conduct a Management Review 25
- **Review Procedures**
  - Planning 26
  - Preparation and Execution 26
- **Technical Reviews**
  - Responsibilities 27
  - Input 27
- **Inspections**
  - Responsibilities 28
  - Inspection Rules and Procedures 29
- **Walkthroughs**
  - Responsibilities 29
- **Audits**
  - 30
  - 32

## Chapter 2 Management and Process

- **Introduction** 35
- **Part 1: Software Management**
  - 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 Integration of IEEE 12207 and SESC 39
  - Acquisition 39
  - Development 39
  - Operation 40
  - Supporting Documentation 40
- **Part 2: Software Life Cycle Models**
  - 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**
  - Primary Life Cycle Process 45
  - Acquisition Process 46
  - Supply Process 48
Section II Testing

Chapter 3 Testing: Concept and Definition

Introduction

Part 1: Testing in the Software Life Cycle
What Is Software Testing?
Requirements
Identification and Specification
Specification
Functional System Development
Technical System Design
Component Specification
Coding
Testing
Are We Building the Right System?
Are We Building the System Right?

Part 2: Software Testing Life Cycle
SDLC and STLC

Part 3: Kinds/Types of Testing
Black Box Testing
White Box Testing
Unit Testing
Integration Testing
Incremental Integration Testing
Functional Testing
System Testing
End-to-End Testing
Sanity Testing
Regression Testing
Acceptance Testing
Load Testing
Stress Testing
Performance Testing
Usability Testing
Install/Uninstall Testing
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

Chapter 5 Test: Execution and Reporting 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

Software Quality Assurance: Integrating Testing, Security, and Audit
https://www.crcpress.com/9781498735537
SOFTWARE QUALITY ASSURANCE: INTEGRATING TESTING, SECURITY, AND AUDIT

SECTION III  CHALLENGES

CHAPTER 6  INCIDENT MANAGEMENT

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

Software Quality Assurance: Integrating Testing, Security, and Audit
https://www.crcpress.com/9781498735537
<table>
<thead>
<tr>
<th>Chapter 7</th>
<th>Defect Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>145</td>
</tr>
<tr>
<td>Part 1: Definition and Analysis</td>
<td>145</td>
</tr>
<tr>
<td>Definitions</td>
<td>145</td>
</tr>
<tr>
<td>Defect</td>
<td>145</td>
</tr>
<tr>
<td>Definition of an Error</td>
<td>146</td>
</tr>
<tr>
<td>Defect Repository</td>
<td>146</td>
</tr>
<tr>
<td>What Causes Defects in Software</td>
<td>146</td>
</tr>
<tr>
<td>Detecting a Defect Early</td>
<td>148</td>
</tr>
<tr>
<td>What Is the Cost of Defects Not Being Detected Early?</td>
<td>148</td>
</tr>
<tr>
<td>Defect Life Cycle Steps</td>
<td>150</td>
</tr>
<tr>
<td>Step 1: Recognition or Identification</td>
<td>150</td>
</tr>
<tr>
<td>Step 2: Investigation</td>
<td>150</td>
</tr>
<tr>
<td>Step 3: Action</td>
<td>150</td>
</tr>
<tr>
<td>Step 4: Disposition</td>
<td>151</td>
</tr>
<tr>
<td>Objectives of Testing</td>
<td>151</td>
</tr>
<tr>
<td>Reduce the Risk of Failure</td>
<td>151</td>
</tr>
<tr>
<td>Reduce the Cost of Testing</td>
<td>151</td>
</tr>
</tbody>
</table>
## CONTENTS

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 154
   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

Software Quality Assurance: Integrating Testing, Security, and Audit
https://www.crcpress.com/9781498735537
## Chapter 8 Risk, Vulnerability, and Threat Management

### Introduction

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part 1: Risk Management</td>
<td>171</td>
</tr>
<tr>
<td>Types of Risks</td>
<td>172</td>
</tr>
<tr>
<td>Impact of Risk</td>
<td>173</td>
</tr>
<tr>
<td>Dealing with Risk</td>
<td>173</td>
</tr>
<tr>
<td>Risk Management Life Cycle</td>
<td>174</td>
</tr>
<tr>
<td>Risk Identification</td>
<td>174</td>
</tr>
<tr>
<td>Ten Effective Methods to Identify Risks</td>
<td>174</td>
</tr>
<tr>
<td>Brainstorming</td>
<td>174</td>
</tr>
<tr>
<td>Survey</td>
<td>175</td>
</tr>
<tr>
<td>Interview</td>
<td>175</td>
</tr>
<tr>
<td>Practical Experience and Understanding</td>
<td>175</td>
</tr>
<tr>
<td>Research</td>
<td>176</td>
</tr>
<tr>
<td>Potential Risk Lists</td>
<td>176</td>
</tr>
<tr>
<td>Lessons Learned</td>
<td>176</td>
</tr>
<tr>
<td>Risk-Oriented Analysis</td>
<td>176</td>
</tr>
<tr>
<td>Design Template</td>
<td>176</td>
</tr>
<tr>
<td>Risk Assessment</td>
<td>176</td>
</tr>
<tr>
<td>What Is Risk Assessment?</td>
<td>177</td>
</tr>
<tr>
<td>Risk Assessment Process</td>
<td>177</td>
</tr>
<tr>
<td>Risk Assessment Involves Identified Risks</td>
<td>178</td>
</tr>
<tr>
<td>Technology Risk Assessment and Mitigation (TRAM) (Sample)</td>
<td>178</td>
</tr>
<tr>
<td>Business Risk</td>
<td>178</td>
</tr>
<tr>
<td>Catastrophic (A)</td>
<td>179</td>
</tr>
<tr>
<td>Critical (B)</td>
<td>179</td>
</tr>
<tr>
<td>Moderate (C)</td>
<td>179</td>
</tr>
<tr>
<td>Minor (D)</td>
<td>179</td>
</tr>
<tr>
<td>Risk Assessment Matrix</td>
<td>181</td>
</tr>
<tr>
<td>Negligible (E)</td>
<td>181</td>
</tr>
<tr>
<td>Risk Response</td>
<td>181</td>
</tr>
<tr>
<td>Avoid</td>
<td>181</td>
</tr>
<tr>
<td>Transfer</td>
<td>181</td>
</tr>
<tr>
<td>Reduce</td>
<td>182</td>
</tr>
<tr>
<td>Accept</td>
<td>182</td>
</tr>
<tr>
<td>Risk Mitigation</td>
<td>182</td>
</tr>
<tr>
<td>Risk Contingency Plan</td>
<td>183</td>
</tr>
<tr>
<td>Technology Contingency Plan (TCP) (Sample)</td>
<td>184</td>
</tr>
</tbody>
</table>
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Risk Questionnaire (ARQ)</td>
<td>184</td>
</tr>
<tr>
<td>Project Risk Log</td>
<td>184</td>
</tr>
<tr>
<td>Part 2: Vulnerability, Risk, and Threat Analysis</td>
<td>184</td>
</tr>
<tr>
<td>Vulnerability and Risk</td>
<td>185</td>
</tr>
<tr>
<td>Step 1: Determine What Is Being Protected and Why</td>
<td>185</td>
</tr>
<tr>
<td>Sample Statement</td>
<td>186</td>
</tr>
<tr>
<td>Step 2: Identify the System</td>
<td>186</td>
</tr>
<tr>
<td>Step 3: Characterize System Operations</td>
<td>187</td>
</tr>
<tr>
<td>Step 4: Ascertain What One Does and Does Not Have Control Over</td>
<td>187</td>
</tr>
<tr>
<td>Vulnerability and Threat Definitions</td>
<td>188</td>
</tr>
<tr>
<td>Definitions</td>
<td>188</td>
</tr>
<tr>
<td>Four Levels of Threats</td>
<td>188</td>
</tr>
<tr>
<td>Four Steps of Risk Assessment</td>
<td>189</td>
</tr>
<tr>
<td>Step 1: Analysis Techniques Are Selected and Used</td>
<td>189</td>
</tr>
<tr>
<td>Step 2: Identify Vulnerabilities, Their Type, Source, and Severity</td>
<td>190</td>
</tr>
<tr>
<td>Step 3: Identify Threats, Their Type, Source, and Likelihood</td>
<td>190</td>
</tr>
<tr>
<td>Step 4: Evaluate Transaction Paths, Threat Zones, and Risk Exposure</td>
<td>190</td>
</tr>
<tr>
<td>Part 3: OCTAVE and Risk Management</td>
<td>192</td>
</tr>
<tr>
<td>What Is OCTAVE?</td>
<td>192</td>
</tr>
<tr>
<td>OCTAVE Phases</td>
<td>194</td>
</tr>
<tr>
<td>Phase 1: Build Asset-Based Threat Profiles</td>
<td>194</td>
</tr>
<tr>
<td>Phase 2: Identify Infrastructure Vulnerabilities</td>
<td>195</td>
</tr>
<tr>
<td>Phase 3: Develop Security Strategy and Plans</td>
<td>195</td>
</tr>
<tr>
<td>OCTAVE Way of Risk Management</td>
<td>195</td>
</tr>
<tr>
<td>OCTAVE in Risk Management</td>
<td>196</td>
</tr>
<tr>
<td>Appendix A—Sample</td>
<td>197</td>
</tr>
<tr>
<td>Vulnerability/Risk Assessment</td>
<td>197</td>
</tr>
<tr>
<td>For Pharmacy Handheld Technology</td>
<td>197</td>
</tr>
<tr>
<td>Introduction</td>
<td>197</td>
</tr>
<tr>
<td>Statement of Goals</td>
<td>197</td>
</tr>
<tr>
<td>High-Level System Entity Control Analysis</td>
<td>197</td>
</tr>
<tr>
<td>Vulnerability and Threat Analysis</td>
<td>200</td>
</tr>
<tr>
<td>Physical Structure</td>
<td>202</td>
</tr>
<tr>
<td>Virtual Private Network as a Risk</td>
<td>202</td>
</tr>
<tr>
<td>The Major Strengths of Utilizing Internet-Based VPN Services</td>
<td>203</td>
</tr>
<tr>
<td>Assumptions</td>
<td>203</td>
</tr>
<tr>
<td>Appendix B</td>
<td>204</td>
</tr>
<tr>
<td>Risk Factors Assumptions</td>
<td>204</td>
</tr>
<tr>
<td>Investment Size</td>
<td>204</td>
</tr>
<tr>
<td>Management Process Maturity</td>
<td>205</td>
</tr>
</tbody>
</table>
Part 1: Definition and Importance
- What Is Information Security?
- Difference between Privacy and Security
- Key Points on Information Security
- From What Threats Does Information Need to Be Secured?
  - Cybercrime
  - Types of Cybercrime
  - Computer Virus
  - Scam
  - Money Laundering
  - Phishing
- What Kind of Information Needs to Be Secured
- Some Examples of Recent Phishing
- Identity Theft
  - Information That Is Considered Identity
    - Social Security Numbers
    - Date of Birth
    - Current and Previous Addresses and Phone Numbers
  - Current and Previous Employment Information
  - Financial Account Information
  - Mother's Maiden Name
    - Other Personal Information
  - Password for Nonfinancial Accounts
  - Password for Financial Accounts
  - Criminal Activities That Lead to Cybercrime
    - Spyware
- Objective of Information Security
- Why Is Security Important?
- What Is the Benefit of Information Security?

Part 2: Methodology
- The Strategy
- Security Standards
  - ISO 15408
  - Control Objectives for Information and (Related) Technology (COBIT)
  - ISO 17799/BS7799
  - COBIT
  - OCTAVE
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

Software Quality Assurance: Integrating Testing, Security, and Audit
https://www.crcpress.com/9781498735537
## CONTENTS

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

Software Quality Assurance: Integrating Testing, Security, and Audit
https://www.crcpress.com/9781498735537
## CONTENTS

- 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
XX

CONTENTS

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

Chapter 10 Information Audit 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

Software Quality Assurance: Integrating Testing, Security, and Audit
https://www.crcpress.com/9781498735537
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auditor</td>
<td>286</td>
</tr>
<tr>
<td>Initiator</td>
<td>286</td>
</tr>
<tr>
<td>Audited Organization</td>
<td>286</td>
</tr>
<tr>
<td>Auditor's Qualifications</td>
<td>286</td>
</tr>
<tr>
<td>Choosing an Auditor</td>
<td>286</td>
</tr>
<tr>
<td>Auditor's Education</td>
<td>287</td>
</tr>
<tr>
<td>Knowledge and Skills</td>
<td>287</td>
</tr>
<tr>
<td>Experience</td>
<td>288</td>
</tr>
<tr>
<td>Knowledge</td>
<td>288</td>
</tr>
<tr>
<td>Talent</td>
<td>288</td>
</tr>
<tr>
<td>Competence</td>
<td>288</td>
</tr>
<tr>
<td><strong>Part 2: Audit Process and Procedure</strong></td>
<td>288</td>
</tr>
<tr>
<td>Audit Process</td>
<td>289</td>
</tr>
<tr>
<td>Audit Process Implementation</td>
<td>290</td>
</tr>
<tr>
<td>Support for the Audit Process</td>
<td>290</td>
</tr>
<tr>
<td>Procedures</td>
<td>290</td>
</tr>
<tr>
<td>Management Preparation</td>
<td>290</td>
</tr>
<tr>
<td>Verification of Quality Manual</td>
<td>291</td>
</tr>
<tr>
<td>Verification of Implementation of the Quality Manual</td>
<td>291</td>
</tr>
<tr>
<td>Sample Work Instructions</td>
<td>293</td>
</tr>
<tr>
<td>Postimplementation Review</td>
<td>293</td>
</tr>
<tr>
<td>Key Phase Review</td>
<td>294</td>
</tr>
<tr>
<td>Project Management Methodology Assessment</td>
<td>294</td>
</tr>
<tr>
<td>Privacy and Audit Management</td>
<td>295</td>
</tr>
<tr>
<td>Five Key Focus Areas for Project Audits</td>
<td>295</td>
</tr>
<tr>
<td>Business and IT Alignment</td>
<td>296</td>
</tr>
<tr>
<td>Project Management</td>
<td>296</td>
</tr>
<tr>
<td>IT Solution Readiness</td>
<td>297</td>
</tr>
<tr>
<td>Solution Design</td>
<td>297</td>
</tr>
<tr>
<td>Organizational and Process Change Management</td>
<td>297</td>
</tr>
<tr>
<td>The Audit Report</td>
<td>298</td>
</tr>
<tr>
<td><strong>Part 3: Auditing and Information Security</strong></td>
<td>299</td>
</tr>
<tr>
<td>Defined and Planned Strategy</td>
<td>299</td>
</tr>
<tr>
<td>Auditing Privacy Risks</td>
<td>299</td>
</tr>
<tr>
<td>Auditing Data Categorization</td>
<td>300</td>
</tr>
<tr>
<td>Auditing Law and Regulation Aspects</td>
<td>301</td>
</tr>
<tr>
<td>Organization Threats</td>
<td>301</td>
</tr>
<tr>
<td>Application Risks</td>
<td>301</td>
</tr>
<tr>
<td>Business Process Risks</td>
<td>302</td>
</tr>
<tr>
<td>Auditing IT Vulnerabilities</td>
<td>302</td>
</tr>
<tr>
<td>Identifying Insignificant Vulnerability Management</td>
<td>302</td>
</tr>
<tr>
<td>The Internal Auditor’s Role About Information Security</td>
<td>303</td>
</tr>
<tr>
<td>Vulnerability and Risk</td>
<td>303</td>
</tr>
<tr>
<td>Persistent Auditing and Monitoring</td>
<td>304</td>
</tr>
<tr>
<td>Suggested Readings</td>
<td>305</td>
</tr>
</tbody>
</table>
Chapter 11 Software Reliability and Process Improvement

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

Software Quality Assurance: Integrating Testing, Security, and Audit
https://www.crcpress.com/9781498735537
### CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Making Reliability Metrics Meaningful</td>
<td>318</td>
</tr>
<tr>
<td>Standards Defining Software Measurement</td>
<td>318</td>
</tr>
<tr>
<td><strong>Productivity Metrics: IEEE 1045</strong></td>
<td>319</td>
</tr>
<tr>
<td><strong>Software Reliability: IEEE 982</strong></td>
<td>319</td>
</tr>
<tr>
<td>Quality Metrics Methodology</td>
<td>320</td>
</tr>
<tr>
<td><strong>IEEE 1061–1992</strong></td>
<td>320</td>
</tr>
<tr>
<td>Software Reliability Measurement</td>
<td>321</td>
</tr>
<tr>
<td>What Is a Model?</td>
<td>321</td>
</tr>
<tr>
<td>Qualities of a Good Model</td>
<td>321</td>
</tr>
<tr>
<td>The Importance of Data</td>
<td>321</td>
</tr>
<tr>
<td>Metrics and Models</td>
<td>322</td>
</tr>
<tr>
<td>Model Development and Independent Metrics</td>
<td>322</td>
</tr>
<tr>
<td>The Issue of Availability</td>
<td>322</td>
</tr>
<tr>
<td>Data Retention and Use</td>
<td>322</td>
</tr>
<tr>
<td>Validity</td>
<td>323</td>
</tr>
<tr>
<td>Software Reliability Estimation</td>
<td>323</td>
</tr>
<tr>
<td>CMMs: The Software Engineering Institute’s Capability Maturity Model</td>
<td>323</td>
</tr>
<tr>
<td>Maturity Levels</td>
<td>323</td>
</tr>
<tr>
<td>Initial</td>
<td>324</td>
</tr>
<tr>
<td>Repeatable</td>
<td>324</td>
</tr>
<tr>
<td>Defined</td>
<td>324</td>
</tr>
<tr>
<td>Managed</td>
<td>324</td>
</tr>
<tr>
<td>Optimized</td>
<td>324</td>
</tr>
<tr>
<td>Common Features</td>
<td>325</td>
</tr>
<tr>
<td>CMMI</td>
<td>325</td>
</tr>
<tr>
<td>Staged Representation</td>
<td>325</td>
</tr>
<tr>
<td>Continuous Representation</td>
<td>325</td>
</tr>
<tr>
<td>Disciplines and Environments</td>
<td>326</td>
</tr>
<tr>
<td>CMMI Application</td>
<td>326</td>
</tr>
<tr>
<td>Maturity Levels</td>
<td>326</td>
</tr>
<tr>
<td>Process Areas</td>
<td>326</td>
</tr>
<tr>
<td>Level Three Process Areas</td>
<td>327</td>
</tr>
<tr>
<td>Level Four Process Areas</td>
<td>327</td>
</tr>
<tr>
<td>Level Five Process Areas</td>
<td>327</td>
</tr>
<tr>
<td>IDEAL</td>
<td>327</td>
</tr>
</tbody>
</table>

#### Part 2: Software Process Improvement and Capability Determination (SPICE)

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO 15504 and Management</td>
<td>328</td>
</tr>
<tr>
<td>The Assessment Process</td>
<td>328</td>
</tr>
<tr>
<td>The Reference Model</td>
<td>328</td>
</tr>
<tr>
<td>The Capability Dimension</td>
<td>329</td>
</tr>
<tr>
<td>The Engineering Process Category</td>
<td>329</td>
</tr>
<tr>
<td>The Project Process Category</td>
<td>330</td>
</tr>
<tr>
<td>The Support Process Category</td>
<td>330</td>
</tr>
<tr>
<td>The Organization Process Category</td>
<td>330</td>
</tr>
<tr>
<td>ISO/IEC 15288 Processes</td>
<td>330</td>
</tr>
</tbody>
</table>
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

---

† 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).

```c
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

Software Quality Assurance: Integrating Testing, Security, and Audit
https://www.crcpress.com/9781498735537
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,

![Defect life cycle from new to closed.](image-url)
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**

<table>
<thead>
<tr>
<th>VERSION CHANGE DATE</th>
<th>WHAT KIND OF CHANGE/REVISION</th>
<th>WHERE CHANGED HAPPENED (SECTION/PAGE)</th>
<th>REVISED BY NAME AND TITLE</th>
<th>APPROVED BY NAME AND TITLE</th>
</tr>
</thead>
</table>

**Documentation**

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

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

(Continued)
### Defect States

The mandatory states of a defect are
- Submitted/New
- Open
- Resolved
- Closed
- Canceled
- Deferred

### Table: Defect States

<table>
<thead>
<tr>
<th>STEP #</th>
<th>STATUS</th>
<th>DESCRIPTION</th>
<th>PRIMARY PERFORMER</th>
<th>OUTPUT/EVIDENCE</th>
<th>INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>4.</strong></td>
<td><strong>REOPENING THE DEFECT</strong></td>
<td>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.</td>
<td>Test team Business team</td>
<td>Defect record is moved to a “pending resolution” state in the defect repository.</td>
<td></td>
</tr>
<tr>
<td><strong>5.</strong></td>
<td><strong>CLOSING THE DEFECT</strong></td>
<td>The submitter retests the resolved defect and the defect does not exist. Close the resolved defect.</td>
<td>Test team Business team</td>
<td>Defect record is moved to “closed” state in the defect repository.</td>
<td></td>
</tr>
<tr>
<td><strong>6.</strong></td>
<td><strong>DEFERRING THE DEFECT</strong></td>
<td>Business team determined to defer the acknowledged defect.</td>
<td>Business team</td>
<td>Defect record is moved to “deferred” state in the defect repository.</td>
<td></td>
</tr>
<tr>
<td><strong>7.</strong></td>
<td><strong>CANCELING THE DEFECT</strong></td>
<td>Business or development team cancels the acknowledged or deferred defect. Submitter concurs.</td>
<td>Test team Business team</td>
<td>Defect record is moved to “cancelled” state in the defect repository.</td>
<td></td>
</tr>
<tr>
<td>ATTRIBUTES</td>
<td>DESCRIPTION</td>
<td>TYPE</td>
<td>BUSINESS RULES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------------</td>
<td>-------------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Defect ID</td>
<td>Defect name of ID</td>
<td>List</td>
<td>Name or ID of the project. Most recent project name or ID must be used if a defect is found in production.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Optional (O)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Required (R)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Conditionally Required (CR)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project ID</td>
<td>Project name or ID</td>
<td>List</td>
<td>Unique application identifier. Should match to the CMDB and asset ID databases.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Optional (O)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Required (R)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Conditionally Required (CR)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application</td>
<td>System in which defect was identified</td>
<td>System List</td>
<td>List of functional areas, modules, or components. The defect is associated with an application.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Optional (O)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Required (R)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Conditionally Required (CR)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functional area</td>
<td>Module/ subsystem/ component in which defect was identified</td>
<td>System List</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Optional (O)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Required (R)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Conditionally Required (CR)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Headline</td>
<td>One line summary of the defect</td>
<td>Text</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Optional (O)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Required (R)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Detailed description of problem that includes steps to reproduce the defect, actual results, and expected results</td>
<td>Text</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Optional (O)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Required (R)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDLC phase found</td>
<td>Phase where the defect was detected</td>
<td>List</td>
<td>Analysis (optional), design (optional), construction, system test, user test, implementation and postimplementation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Optional (O)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Required (R)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Conditionally Required (CR)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Found in environment</td>
<td>Environment in which defect was found</td>
<td>List</td>
<td>Development, test, acceptance, production, integration and contingency. Not required if root cause is requirement or design.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Optional (O)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Required (R)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Conditionally Required (CR)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Found in release number</td>
<td>Release number in which the defect is found</td>
<td>System List</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Optional (O)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Required (R)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Conditionally Required (CR)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(Continued)
### Defect Attributes

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

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Description</th>
<th>Type</th>
<th>Optional (O)</th>
<th>Required (R)</th>
<th>Conditionally Required (CR)</th>
<th>Business Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Closed in release number</td>
<td>Release number for which the defect is closed</td>
<td>System list</td>
<td>CR</td>
<td>0</td>
<td>1</td>
<td>Conditionally required if the state is closed</td>
</tr>
<tr>
<td>Remedy incident ID</td>
<td>Problem ticket number for defects found in production</td>
<td>Text</td>
<td>CR</td>
<td>0</td>
<td>1</td>
<td>Conditionally required if the SDLC phase found is implementation or postimplementation</td>
</tr>
<tr>
<td>Test case ID</td>
<td>Associated test case ID</td>
<td>Text</td>
<td>CR</td>
<td>0</td>
<td>1</td>
<td>Not required if root cause is requirement or design</td>
</tr>
<tr>
<td>Functionality type</td>
<td>Lists the type of functionality that introduced the defect</td>
<td>List</td>
<td>R</td>
<td>0</td>
<td>1</td>
<td>New functionality, existing functionality</td>
</tr>
<tr>
<td>Severity</td>
<td>Impacts to application functionalities, business processes, or interfaces causing minor to critical disruption to application usage</td>
<td>List</td>
<td>R</td>
<td>0</td>
<td>1</td>
<td>Severity of the defect must be set in consensus with stakeholders to one of the following:</td>
</tr>
</tbody>
</table>

(Continued)

Priorities will be set by the submitter of the defect to one of the following:

- Critical
- High
- Medium
- Low

(Continued)
## DEFECT MANAGEMENT

<table>
<thead>
<tr>
<th>ATTRIBUTES</th>
<th>DESCRIPTION</th>
<th>TYPE</th>
<th>BUSINESS RULES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root cause</td>
<td>Analysis of what caused the defect</td>
<td>List</td>
<td>Required if root cause is:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R</td>
<td>Requirements</td>
</tr>
<tr>
<td>Root cause reason</td>
<td>Provide the reason for the root cause of the issue.</td>
<td>List</td>
<td>Incomplete/missing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CR</td>
<td>Unclear</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Inconsistent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Incorrect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Not traceable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Not testable</td>
</tr>
<tr>
<td>Resolution notes</td>
<td>Details regarding the resolution of defect</td>
<td>Text</td>
<td>Conditionally required if the state is not submitted</td>
</tr>
<tr>
<td>State</td>
<td>Provides a current state of the defect's current flow while going through defect resolution process</td>
<td>System action</td>
<td>This is the defect state at any point in time. The values in this field are auto populated</td>
</tr>
<tr>
<td>Deferral Business Impact</td>
<td>Business impact description if defect is deferred as well as work around if applicable</td>
<td>Text</td>
<td>Required if the defect is deferred</td>
</tr>
<tr>
<td>Submitter</td>
<td>Name of person that created the defect</td>
<td>System generated</td>
<td>Name of the person that last change the state of the defect. Required if the state of the defect is deferred, closed, or canceled.</td>
</tr>
<tr>
<td>State updated By</td>
<td>Name of person that last changed the state of the defect</td>
<td>System list or system generated</td>
<td></td>
</tr>
<tr>
<td>Update date</td>
<td>Date on which defect state</td>
<td>System</td>
<td>Defect repository will maintain the audit trail for the defect</td>
</tr>
<tr>
<td>Workaround</td>
<td>Lists the work around for deferred defects</td>
<td>Text</td>
<td>Required if the state of the defect is deferred</td>
</tr>
<tr>
<td>System test</td>
<td>Identifies whether the defect was leaked from system test</td>
<td>List</td>
<td>Required if the SDLC phase found is user test of higher</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CR</td>
<td></td>
</tr>
</tbody>
</table>

Software Quality Assurance: Integrating Testing, Security, and Audit
https://www.crcpress.com/9781498735537
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.

<table>
<thead>
<tr>
<th>PRIORITY DESCRIPTION</th>
<th>PRIORITY DEFINITIONS FOR DEFECTS IN NONPRODUCTION</th>
<th>PRIORITY DEFINITIONS FOR PRODUCTION DEFECTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical</td>
<td>Immediate attention—critical may also mean that it might be blocking some other activities. The critical defect should be resolved immediately.</td>
<td>Immediate attention—must receive highest development priority and should be resolved immediately.</td>
</tr>
<tr>
<td>High</td>
<td>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.</td>
<td>Should be reported immediately to the development team. A response or action plan must be provided within 2 working days.</td>
</tr>
<tr>
<td>Medium</td>
<td>A response or action plan should be reported within 5 working days.</td>
<td>A response or action plan should be reported within 5 working days. This defect should be resolved in the next release.</td>
</tr>
<tr>
<td>Low</td>
<td>Fix dates are subject to negotiation. An action plan before the next release.</td>
<td>Fix dates are subject to negotiation. An action plan before the next release.</td>
</tr>
</tbody>
</table>

Defect Severeities

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.

<table>
<thead>
<tr>
<th>SEVERITY DESCRIPTION</th>
<th>SEVERITY DEFINITIONS FOR SYSTEM, USER, AND PRODUCTION</th>
<th>SEVERITY DEFINITIONS FOR REQUIREMENT DEFECTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical</td>
<td>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.</td>
<td>The reason for the requirement defect could be considered such as Incomplete/missing Inconsistent Incorrect</td>
</tr>
</tbody>
</table>

(Continued)
### SEVERITY DESCRIPTION

<table>
<thead>
<tr>
<th>SEVERITY</th>
<th>SEVERITY DEFINITIONS FOR SYSTEM, USER, AND PRODUCTION</th>
<th>SEVERITY DEFINITIONS FOR REQUIREMENT DEFECTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>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.</td>
<td>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</td>
</tr>
<tr>
<td>Medium</td>
<td>Minor business disruption but has a workaround, minor usability issues. This defect should be resolved before exiting current phase or releasing to production.</td>
<td>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</td>
</tr>
<tr>
<td>Low</td>
<td>The defect may be cosmetic in nature or a usability annoyance, such as warning messages, misspelled words, etc.</td>
<td>Formatting or organizational observation or a grammatical or spelling error not affecting the meaning. The reason for the requirement defect is usually unclear</td>
</tr>
</tbody>
</table>

## 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).
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 the 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),

<table>
<thead>
<tr>
<th>METRICS DATA ELEMENT</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planned progression coverage</td>
<td>100%</td>
</tr>
<tr>
<td>Prerelease overall regression capability</td>
<td>90%</td>
</tr>
<tr>
<td>Prerelease overall regression automation</td>
<td>70%</td>
</tr>
<tr>
<td>Planned regression coverage</td>
<td>100%</td>
</tr>
<tr>
<td>Planned regression coverage—automated</td>
<td>40%</td>
</tr>
<tr>
<td>Expected initial pass rate</td>
<td>90%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>METRICS DATA ELEMENT</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual progression coverage</td>
<td>100%</td>
</tr>
<tr>
<td>Progression coverage—automated</td>
<td>50%</td>
</tr>
<tr>
<td>Actual regression coverage</td>
<td>80%</td>
</tr>
<tr>
<td>Postrelease overall regression capability</td>
<td></td>
</tr>
<tr>
<td>Postrelease overall regression automation</td>
<td></td>
</tr>
<tr>
<td>Actual initial pass rate</td>
<td>80%</td>
</tr>
<tr>
<td>Final pass rate</td>
<td>100%</td>
</tr>
<tr>
<td>System test defects—critical/high</td>
<td>5</td>
</tr>
<tr>
<td>System test defects—medium/low</td>
<td>12</td>
</tr>
<tr>
<td>System test defects—deferred</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>METRICS DATA ELEMENT</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>UAT defects—critical/high</td>
<td>2</td>
</tr>
<tr>
<td>UAT defects—medium/low</td>
<td>3</td>
</tr>
<tr>
<td>UAT defects—deferred</td>
<td>1</td>
</tr>
<tr>
<td>System test leakage—critical/high</td>
<td>0</td>
</tr>
<tr>
<td>System test leakage—medium/low</td>
<td>1</td>
</tr>
</tbody>
</table>
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).

<table>
<thead>
<tr>
<th>DEFECT ID</th>
<th>STATE</th>
<th>SDLC PHASE</th>
<th>ROOT CAUSE</th>
<th>DEFECT PHASE AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>Closed</td>
<td>System Test</td>
<td>Requirement</td>
<td>3</td>
</tr>
<tr>
<td>1002</td>
<td>Closed</td>
<td>Design</td>
<td>Code</td>
<td>-1</td>
</tr>
<tr>
<td>1003</td>
<td>Closed</td>
<td>System Test</td>
<td>Design</td>
<td>2</td>
</tr>
<tr>
<td>1004</td>
<td>Closed</td>
<td>System Test</td>
<td>Requirement</td>
<td>3</td>
</tr>
<tr>
<td>1005</td>
<td>Closed</td>
<td>System Test</td>
<td>Code</td>
<td>1</td>
</tr>
<tr>
<td>1006</td>
<td>Deferred</td>
<td>System Test</td>
<td>Code</td>
<td>1</td>
</tr>
<tr>
<td>1007</td>
<td>Deferred</td>
<td>User Test</td>
<td>Code</td>
<td>2</td>
</tr>
<tr>
<td>1008</td>
<td>Closed</td>
<td>Design</td>
<td>Code</td>
<td>-1</td>
</tr>
<tr>
<td>1009</td>
<td>Closed</td>
<td>Design</td>
<td>Requirement</td>
<td>1</td>
</tr>
<tr>
<td>1010</td>
<td>Closed</td>
<td>System Test</td>
<td>Code</td>
<td>1</td>
</tr>
<tr>
<td>1011</td>
<td>Closed</td>
<td>User Test</td>
<td>Design</td>
<td>3</td>
</tr>
<tr>
<td>1012</td>
<td>Deferred</td>
<td>User Test</td>
<td>Code</td>
<td>4</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Defect Type</th>
<th>% Pass Rate</th>
<th># of Defects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expected initial pass rate</td>
<td>90%</td>
<td></td>
</tr>
<tr>
<td>Actual initial pass rate</td>
<td>80%</td>
<td></td>
</tr>
<tr>
<td>UAT - deferred defects</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>SIT - deferred defects</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>SIT - leakage (critical/high)</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>System test defects</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Deferred defects</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Critical/high</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Medium/low</td>
<td></td>
<td></td>
</tr>
<tr>
<td>High/critical</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Test execution and defect**

- **Pass rate**: 90% (Actual) 80% (Expected)
- **UAT - deferred defects**: 5
- **SIT - deferred defects**: 2
- **SIT - leakage (critical/high)**: 12
- **System test defects**: 0
- **Deferred defects**: 1
- **Critical/high**: 2
- **Medium/low**: 0
- **High/critical**: 0
Defect root cause analysis

Average defect phase age = 1.73/5
Calculation = (Phase detected - Phase injected)

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.