

CONTENTS

PREFACE	xix
Features and Organization / xxi	
Practice Descriptions / xxiii	
Intended Audience / xxiv	
Acknowledgments / xxiv	
Permissions / xxvi	
Disclaimer / xxvi	
1 THE CASE FOR AUTOMATED DEFECT PREVENTION	1
1.1 What Is ADP? / 1	
1.2 What Are the Goals of ADP? / 3	
1.2.1 People: Stimulated and Satisfied / 3	
1.2.2 Product: High Quality / 4	
1.2.3 Organization: Increased Productivity and Operational Efficiency / 5	
1.2.4 Process: Controlled, Improved, and Sustainable / 6	
1.2.5 Project: Managed through Informed Decision Making / 7	
1.3 How Is ADP Implemented? / 8	
1.3.1 Principles / 8	
1.3.2 Practices / 9	
1.3.3 Policies / 9	

- 1.3.4 Defect Prevention Mindset / 10
- 1.3.5 Automation / 11
- 1.4 From the Waterfall to Modern Software Development Process Models / 11
- 1.5 Acronyms / 13
- 1.6 Glossary / 13
- 1.7 References / 15
- 1.8 Exercises / 16

2 PRINCIPLES OF AUTOMATED DEFECT PREVENTION 19

- 2.1 Introduction / 19
- 2.2 Defect Prevention: Definition and Benefits / 21
- 2.3 Historical Perspective: Defect Analysis and Prevention in the Auto Industry—What Happened to Deming? / 24
- 2.4 Principles of Automated Defect Prevention / 26
 - 2.4.1 Principle 1—Establishment of Infrastructure: “Build a Strong Foundation through Integration of People and Technology” / 26
 - 2.4.2 Principle 2—Application of General Best Practices: “Learn from Others’ Mistakes” / 28
 - 2.4.3 Principle 3—Customization of Best Practices: “Learn from Your Own Mistakes and Improve the Process” / 29
 - 2.4.4 Principle 4—Measurement and Tracking of Project Status: “Understand the Past and Present to Make Decisions about the Future” / 31
 - 2.4.5 Principle 5—Automation: “Let the Computer Do It” / 32
 - 2.4.6 Principle 6—Incremental Implementation of ADP’s Practices and Policies / 36
- 2.5 Automated Defect Prevention–Based Software Development Process Model / 38
- 2.6 Examples / 41
 - 2.6.1 Focus on Root Cause Analysis of a Defect / 41
 - 2.6.2 Focus on Infrastructure / 44
 - 2.6.3 Focus on Customized Best Practice / 45
 - 2.6.4 Focus on Measurements of Project Status / 47
- 2.7 Acronyms / 48
- 2.8 Glossary / 48
- 2.9 References / 50
- 2.10 Exercises / 51

3	INITIAL PLANNING AND INFRASTRUCTURE	53
3.1	Introduction / 53	
3.2	Initial Software Development Plan / 54	
3.2.1	Product / 54	
3.2.2	People / 54	
3.2.3	Technology / 55	
3.2.4	Process / 55	
3.3	Best Practices for Creating People Infrastructure / 56	
3.3.1	Defining Groups / 56	
3.3.2	Determining a Location for Each Group's Infrastructure / 58	
3.3.3	Defining People Roles / 58	
3.3.4	Establishing a Training Program / 61	
3.3.5	Cultivating a Positive Group Culture / 61	
3.4	Best Practices for Creating Technology Infrastructure / 63	
3.4.1	Automated Reporting System / 63	
3.4.2	Policy for Use of Automated Reporting System / 66	
3.4.3	Minimum Technology Infrastructure / 68	
3.4.4	Intermediate Technology Infrastructure / 72	
3.4.5	Expanded Technology Infrastructure / 73	
3.5	Integrating People and Technology / 75	
3.6	Human Factors and Concerns / 77	
3.7	Examples / 78	
3.7.1	Focus on Developer Ideas / 78	
3.7.2	Focus on Reports Generated by the Minimum Infrastructure / 79	
3.8	Acronyms / 80	
3.9	Glossary / 81	
3.10	References / 82	
3.11	Exercises / 83	
4	REQUIREMENTS SPECIFICATION AND MANAGEMENT	85
4.1	Introduction / 85	
4.2	Best Practices for Gathering and Organizing Requirements / 87	
4.2.1	Creating the Product Vision and Scope Document / 87	
4.2.2	Gathering and Organizing Requirements / 89	
4.2.3	Prioritizing Requirements / 93	
4.2.4	Developing Use Cases / 95	
4.2.5	Creating a Prototype to Elicit Requirements / 98	

- 4.2.6 Creating Conceptual Test Cases / 100
- 4.2.7 Requirements Documents Inspection / 101
- 4.2.8 Managing Changing Requirements / 103
- 4.3 Best Practices in Different Environments / 105
 - 4.3.1 Existing versus New Software Project / 105
 - 4.3.2 In-House versus Outsourced Development Teams / 106
- 4.4 Policy for Use of the Requirements Management System / 107
 - 4.4.1 Measurements Related to Requirements Management System / 109
 - 4.4.2 Tracking of Data Related to the Requirements Management System / 110
- 4.5 Examples / 110
 - 4.5.1 Focus on Customized Best Practice / 110
 - 4.5.2 Focus on Monitoring and Managing Requirement Priorities / 112
 - 4.5.3 Focus on Change Requests / 114
- 4.6 Acronyms / 115
- 4.7 Glossary / 115
- 4.8 References / 116
- 4.9 Exercises / 116

5 EXTENDED PLANNING AND INFRASTRUCTURE

119

- 5.1 Introduction / 119
- 5.2 Software Development Plan / 120
- 5.3 Defining Project Objectives / 121
- 5.4 Defining Project Artifacts and Deliverables / 124
 - 5.4.1 The Vision and Scope Document and Project Objectives / 124
 - 5.4.2 SRS, Describing the Product Key Features / 125
 - 5.4.3 Architectural and Detailed Design Documents and Models / 125
 - 5.4.4 List of COTS (Commercial Off-the-Shelf Components) Used / 125
 - 5.4.5 Source and Executable Code / 126
 - 5.4.6 Test Plan / 126
 - 5.4.7 Acceptance Plan / 126
 - 5.4.8 Periodic Reports Generated by the Reporting System / 126
 - 5.4.9 Deployment Plan / 127

- 5.4.10 User and Operational Manuals / 127
- 5.4.11 Customer Training Plan / 127
- 5.5 Selecting a Software Development Process Model / 128
- 5.6 Defining Defect Prevention Process / 129
- 5.7 Managing Risk / 129
- 5.8 Managing Change / 132
- 5.9 Defining Work Breakdown Structure—An Iterative Approach / 132
- 5.10 Best Practices for Estimating Project Effort / 135
 - 5.10.1 Estimation by Using Elements of Wideband Delphi / 137
 - 5.10.2 Estimation by Using Effort Analogy / 138
 - 5.10.3 Estimation by Using Parametric Models / 141
 - 5.10.4 Estimations of Using COTS and Code Reuse / 145
 - 5.10.5 Quality of Estimation and the Iterative Adjustments of Estimates / 145
- 5.11 Best Practices for Preparing the Schedule / 146
- 5.12 Measurement and Tracking for Estimation / 149
- 5.13 Identifying Additional Resource Requirements / 150
 - 5.13.1 Extending the Technology Infrastructure / 151
 - 5.13.2 Extending the People Infrastructure / 156
- 5.14 Examples / 157
 - 5.14.1 Focus on the Root Cause of a Project Scheduling Problem / 157
 - 5.14.2 Focus on Organizing and Tracking Artifacts / 158
 - 5.14.3 Focus on Scheduling and Tracking Milestones / 158
- 5.15 Acronyms / 160
- 5.16 Glossary / 160
- 5.17 References / 162
- 5.18 Exercises / 163

6 ARCHITECTURAL AND DETAILED DESIGN

165

- 6.1 Introduction / 165
- 6.2 Best Practices for Design of System Functionality and Its Quality Attributes / 168
 - 6.2.1 Identifying Critical Attributes of Architectural Design / 168
 - 6.2.2 Defining the Policies for Design of Functional and Nonfunctional Requirements / 172

- 6.2.3 Applying Design Patterns / 175
- 6.2.4 Service-Oriented Architecture / 178
- 6.2.5 Mapping Requirements to Modules / 178
- 6.2.6 Designing Module Interfaces / 181
- 6.2.7 Modeling Modules and Their Interfaces / 182
- 6.2.8 Defining Application Logic / 185
- 6.2.9 Refining Test Cases / 186
- 6.2.10 Design Document Storage and Inspection / 187
- 6.2.11 Managing Changes in Design / 188
- 6.3 Best Practices for Design of Graphical User Interface / 189
 - 6.3.1 Identifying Critical Attributes of User Interface Design / 190
 - 6.3.2 Defining the User Interface Design Policy / 193
 - 6.3.3 Identifying Architectural Patterns Applicable to the User Interface Design / 195
 - 6.3.4 Creating Categories of Actions / 195
 - 6.3.5 Dividing Actions into Screens / 196
 - 6.3.6 Prototyping the Interface / 197
 - 6.3.7 Testing the Interface / 197
- 6.4 Examples / 198
 - 6.4.1 Focus on Module Assignments and Design Progress / 198
 - 6.4.2 Focus on the Number of Use Cases per Module / 198
 - 6.4.3 Focus on Module Implementation Overview / 199
 - 6.4.4 Focus on Customized Best Practice for GUI Design / 199
- 6.5 Acronyms / 200
- 6.6 Glossary / 201
- 6.7 References / 204
- 6.8 Exercises / 205

7 CONSTRUCTION

207

- 7.1 Introduction / 207
- 7.2 Best Practices for Code Construction / 209
 - 7.2.1 Applying Coding Standards throughout Development / 210
 - 7.2.2 Applying the Test-First Approach at the Service and Module Implementation Level / 225
 - 7.2.3 Implementing Service Contracts and/or Module Interfaces before Their Internal Functionality / 226
 - 7.2.4 Applying Test Driven Development for Algorithmically Complex and Critical Code Units / 227

- 7.2.5 Conducting White Box Unit Testing after Implementing Each Unit and before Checking the Code into the Source Control System / 228
- 7.2.6 Verifying Code Consistency with the Requirements and Design / 228
- 7.3 Policy for Use of the Code Source Control System / 229
 - 7.3.1 Measurements Related to Source Control / 234
 - 7.3.2 Tracking of Source Control Data / 236
- 7.4 Policy for Use of Automated Build / 237
 - 7.4.1 Measurements Related to Automated Builds / 240
 - 7.4.2 Tracking of Data Related to Automated Builds / 241
- 7.5 Examples / 241
 - 7.5.1 Focus on a Customized Coding Standard Policy / 241
 - 7.5.2 Focus on Features/Tests Reports / 242
- 7.6 Acronyms / 243
- 7.7 Glossary / 244
- 7.8 References / 245
- 7.9 Exercises / 247

8 TESTING AND DEFECT PREVENTION

249

- 8.1 Introduction / 249
- 8.2 Best Practices for Testing and Code Review / 250
 - 8.2.1 Conducting White Box Unit Testing: Bottom-Up Approach / 251
 - 8.2.2 Conducting Black Box Testing and Verifying the Convergence of Top-Down and Bottom-Up Tests / 256
 - 8.2.3 Conducting Code Reviews as a Testing Activity / 260
 - 8.2.4 Conducting Integration Testing / 263
 - 8.2.5 Conducting System Testing / 265
 - 8.2.6 Conducting Regression Testing / 268
 - 8.2.7 Conducting Acceptance Testing / 270
- 8.3 Defect Analysis and Prevention / 271
- 8.4 Policy for Use of Problem Tracking System / 273
 - 8.4.1 Measurements of Data Related to the Problem Tracking System / 277
 - 8.4.2 Tracking of Data Related to the Problem Tracking System / 277
- 8.5 Policy for Use of Regression Testing System / 278

- 8.5.1 Measurements Related to the Regression Testing System / 279
- 8.5.2 Tracking of Data Related to the Regression Testing System / 279
- 8.6 Examples / 279
 - 8.6.1 Focus on Defect Tracking Reports / 279
 - 8.6.2 Focus on Test Type Reports / 280
 - 8.6.3 Example of a Root Cause Analysis of a Design and Testing Defect / 280
- 8.7 Acronyms / 283
- 8.8 Glossary / 283
- 8.9 References / 285
- 8.10 Exercises / 286

9 TREND ANALYSIS AND DEPLOYMENT

287

- 9.1 Introduction / 287
- 9.2 Trends in Process Control / 288
 - 9.2.1 Process Variations / 288
 - 9.2.2 Process Stabilization / 288
 - 9.2.3 Process Capability / 289
- 9.3 Trends in Project Progress / 290
 - 9.3.1 Analyzing Features/Requirements Implementation Status / 290
 - 9.3.2 Analyzing Source Code Growth / 294
 - 9.3.3 Analyzing Test Results / 296
 - 9.3.4 Analyzing Defects / 299
 - 9.3.5 Analyzing Cost and Schedule / 301
- 9.4 Best Practices for Deployment and Transition / 301
 - 9.4.1 Deployment to a Staging System / 301
 - 9.4.2 Automation of the Deployment Process / 303
 - 9.4.3 Assessing Release Readiness / 304
 - 9.4.4 Release: Deployment to the Production System / 307
 - 9.4.5 Nonintrusive Monitoring / 307
- 9.5 Acronyms / 309
- 9.6 Glossary / 309
- 9.7 References / 309
- 9.8 Exercises / 310

10	MANAGING EXTERNAL FACTORS	311
10.1	Introduction / 311	
10.2	Best Practices for Managing Outsourced Projects / 312	
10.2.1	Establishing a Software Development Outsource Process / 313	
10.2.2	Phase 0: Decision to Outsource / 314	
10.2.3	Phase 1: Planning / 315	
10.2.4	Phase 2: Implementation / 319	
10.2.5	Phase 3: Termination / 321	
10.3	Best Practices for Facilitating IT Regulatory Compliance / 322	
10.3.1	Section 508 of the U.S. Rehabilitation Act / 322	
10.3.2	Sarbanes-Oxley Act of 2002 / 323	
10.4	Best Practices for Implementation of CMMI / 328	
10.4.1	Capability and Maturity Model Integration (CMMI) / 329	
10.4.2	Staged Representation / 330	
10.4.3	Putting Staged Representation–Based Improvement into Practice Using ADP / 330	
10.4.4	Putting Continuous Representation–Based Improvement into Practice Using ADP / 336	
10.5	Acronyms / 337	
10.6	Glossary / 337	
10.7	References / 338	
10.8	Exercises / 339	
11	CASE STUDY: AUTOMATION AS AN AGENT OF CHANGE	341
11.1	Case Study: Implementing Java Coding Standards in a Financial Application / 341	
11.1.1	Company Profile / 341	
11.1.2	Problems / 342	
11.1.3	Solution / 344	
11.1.4	Data Collected / 348	
11.1.5	The Bottom Line Results—Facilitating Change / 351	
11.2	Acronyms / 352	
11.3	Glossary / 352	
11.4	References / 352	

APPENDIX A: A BRIEF SURVEY OF MODERN SOFTWARE DEVELOPMENT PROCESS MODELS	353
A.1 Introduction / 353	
A.2 Rapid Application Development (RAD) and Rapid Prototyping / 353	
A.3 Incremental Development / 355	
A.4 Spiral Model / 355	
A.5 Object-Oriented Unified Process / 357	
A.6 Extreme and Agile Programming / 358	
A.7 References / 359	
APPENDIX B: MARS POLAR LANDER (MPL): LOSS AND LESSONS BY GORDON HEBERT	361
B.1 No Definite Root Cause / 361	
B.2 No Mission Data / 362	
B.3 Root Cause Revisited / 364	
B.4 References / 366	
APPENDIX C: SERVICE-ORIENTED ARCHITECTURE: EXAMPLE OF AN IMPLEMENTATION WITH ADP BEST PRACTICES	369
C.1 Introduction / 369	
C.2 Web Service Creation: Initial Planning and Requirements / 369	
C.2.1 Functional Requirements / 370	
C.2.2 Nonfunctional Requirements / 371	
C.3 Web Service Creation: Extended Planning and Design / 371	
C.3.1 Initial Architecture / 371	
C.3.2 Extended Infrastructure / 373	
C.3.3 Design / 374	
C.4 Web Service Creation: Construction and Testing, Stage 1—Module Implementation / 375	
C.4.1 Applying Coding Standards / 376	
C.4.2 Implementing Interfaces and Applying a Test-First Approach for Modules and Submodules / 376	
C.4.3 Generating White Box JUnit Tests / 377	
C.4.4 Gradually Implementing the Submodule until All Junit Tests Pass and Converge with the Original Black Box Tests / 378	

- C.4.5 Checking Verified Tests into the Source Control System and Running Nightly Regression Tests / 380
- C.5 Web Service Creation: Construction and Testing, Stage 2—The WSDL Document Implementation / 381
 - C.5.1 Creating and Deploying the WSDL Document on the Staging Server as Part of the Nightly Build Process / 381
 - C.5.2 Avoiding Inline Schemas when XML Validation Is Required / 383
 - C.5.3 Avoiding Cyclical Referencing when Using Inline Schemas / 383
 - C.5.4 Verifying WSDL Document for XML Validity / 384
 - C.5.5 Avoiding “Encoded” Coding Style by Checking Interoperability / 384
 - C.5.6 Creating Regression Tests for the WSDL Documents and Schemas to Detect Undesired Changes / 384
- C.6 Web Service Creation: Server Deployment / 384
 - C.6.1 Deploying the Web Service to a Staging Server as Part of the Nightly Build Process / 384
 - C.6.2 Executing Web Service Tests That Verify the Functionality of the Web Service / 385
 - C.6.3 Creating Scenario-Based Tests and Incorporating Them Into the Nightly Test Process / 386
 - C.6.4 Database Testing / 386
- C.7 Web Service Creation: Client Deployment / 386
 - C.7.1 Implementing the Client According to the WSDL Document Specification / 386
 - C.7.2 Using Server Stubs to Test Client Functionality—Deploying the Server Stub as Part of the Nightly Deployment Process / 387
 - C.7.3 Adding Client Tests into the Nightly Test Process / 387
- C.8 Web Service Creation: Verifying Security / 387
 - C.8.1 Determining the Desired Level of Security / 387
 - C.8.2 Deploying Security-Enabled Web Service on Additional Port of the Staging Server / 388
 - C.8.3 Leveraging Existing Tests: Modifying Them to Test for Security and Incorporating Them into the Nightly Test Process / 388
- C.9 Web Service Creation: Verifying Performance through Continuous Performance/Load Testing / 389
 - C.9.1 Starting Load Testing as Early as Possible and Incorporating It into the Nightly Test Process / 389

C.9.2 Using Results of Load Tests to Determine Final
Deployment Configuration / 389

APPENDIX D: AJAX BEST PRACTICE: CONTINUOUS TESTING **391**

D.1 Why AJAX? / 391

D.2 AJAX Development and Testing Challenges / 392

D.3 Continuous Testing / 392

APPENDIX E: SOFTWARE ENGINEERING TOOLS **395**

GLOSSARY **401**

INDEX **415**