More Process Patterns
Managing Object Technology Series

1. What Every Software Manager Must Know to Succeed with Object Technology • John D. Williams
2. Managing Your Move To Object Technology: Guidelines and Strategies for a Smooth Transition • Barry McGibbon
3. The Object Primer: Application Developer’s Guide to Object Orientation • Scott W. Ambler
4. Getting Results with the Object-Oriented Enterprise Model • Thornton Gale and James Eldred
5. Deploying Distributed Business Software • Ted Lewis, Ph.D.
6. The Blueprint for Business Objects • Peter Fingar
7. Object Technology Strategies and Tactics • Gilbert Singer
8. Upgrading Relational Databases with Objects • Robert Vermeulen
10. CUC96: Component-Based Software Engineering • Collected and Introduced by Thomas Jell
11. Developing Business Objects • Edited by Andy Carmichael
12. Component-Based Development for Enterprise Systems: Applying The SELECT Perspective™ • Paul Allen and Stuart Frost
13. Developing Business Systems with CORBA • Waqar Sadiq and Fred Cummins, J.D.
15. Process Patterns: Building Large-Scale Systems Using Object Technology • Scott W. Ambler
17. Enterprise Architectural Patterns: Building Blocks of the Agile Company • Michael A. Beedle
18. No Bull: Object Technology for Executives • William Perlman
19. More Process Patterns: Delivering Large-Scale Systems Using Object Technology • Scott W. Ambler

Additional Volumes in Preparation
To my parents, Bill and Loreen.
About the Author

scottAmbler is an instance of an SeniorOOConsultant with ambySoftInc based in Sharon, Ontario (http://www.ambysoft.com). Messages can be sent to him via the electronic mail contact point scott@ambysoft.com.

scottAmbler is a very versatile object that will change type in order to meet the needs of his clients. For example, he often takes on the role of OOMentor, OOTrainer, OOProcessExpert, or OODEveloper, and columnist with computingCanada (plesmanPublications). Scott has been an instance of an OOConsultant since 1991. scottAmbler has instantiated the book The Object Primer (sigsBooksCambridgeUniversityPress, nyCity, 1995), Building Object Applications That Work (sigsBooksCambridgeUniversityPress, nyCity, 1998), and Process Patterns (sigsBooksCambridgeUniversityPress, nyCity, 1998). He holds the roles of contributing-Editor with softwareDevelopment (millerFreemanPress), columnist with objectMagazine (sigsPublications). scottAmbler is an avid watcher of StarTrekEpisodes, and intends to one day do his doctorate degree at starFleetAcademy.

scottAmbler used to be a MasterStudent object, having received
About the Author

an InformationScienceDegree from universityOfToronto. As a MasterStudent, scottAmbler did a lot of work in OO CASE and instantiated a ThesisPaper object in computer-supported co-operative work (an academic alias for groupWare). Before becoming a MastersStudent, he was an instance of a TechnicalSystemAnalyst at royalBankOfCanada where he originally became interested in object-orientation.
# Contents

*Illustrations*  
*Abbreviations*  
*Foreword*  
*Preface*  

## Chapter 1

### Introduction to More Process Patterns

1.1 Exploring Object-Oriented Myths  
1.2 What Is a Process Pattern?  
1.3 Types of Process Patterns  
1.4 The Object-Oriented Software Process  
1.5 Serial in the Large  
1.5.1 Initiate  
1.5.2 Construct  
1.5.3 Deliver  
1.5.4 Maintain and Support  
1.6 Iterative in the Small  
1.6.1 Initiate
Contents

1.6.2 Define and Validate Initial Requirements 12
1.6.3 Define Initial Management Documents 12
1.6.4 Justify the Project 12
1.6.5 Define the Project Infrastructure 13

1.7 Construct 13
1.7.1 Model 13
1.7.2 Program 14
1.7.3 Generalize 16
1.7.4 Test in the Small 16

1.8 Deliver 17
1.8.1 Test in the Large 18
1.8.2 Rework 18
1.8.3 Release 19
1.8.4 Assess 19

1.9 Maintain and Support 19
1.9.1 Support the Application 19
1.9.2 Identify Defects and Enhancements 20

1.10 Delivering Incremental Releases Over Time 20
1.11 Incremental vs. Parallel Development 22
1.12 With a Little Bit of Glue to Hold It All Together 22
1.13 Quality Assurance 24
1.14 Project Management 25
1.15 People Management 25
1.16 Risk Management 25
1.17 Reuse Management 25
1.18 Training and Education 26
1.19 Metrics Management 26
1.20 Deliverable Management 28
1.21 Infrastructure Management 28
Contents

1.22 The Advantages and Disadvantages of the OOSP 29
1.23 Toward a Mature Object-Oriented Software Process:
The SEI CMM 30
1.24 The Five CMM Maturity Levels 32
1.25 How the Rest of This Book Is Organized 33
1.26 What You have Learned in This Chapter 37
1.27 References and Recommended Reading 38

Part 1 Deliver 41

Chapter 2
The Deliver Phase 43

2.1 Initial Context: Entry Conditions to the Deliver Phase 45
2.2 Solution: How Work Generally Proceeds During the Deliver Phase 46
2.3 Solution: Project Tasks 49
  2.3.1 Managing the Deliver Phase 49
  2.3.2 People Management 50
  2.3.3 Potential Risks During the Deliver Phase 51
  2.3.4 Training and Education Issues 54
2.4 Resulting Context: Exit Conditions from the Deliver Phase 55
2.5 Secrets of Success 56
2.6 Process Checklist 57
2.6 What You Have Learned in This Chapter 58
2.7 References and Recommended Reading 58
Contents

Chapter 3
The Test in the Large Stage

3.1 Initial Context: Entry Conditions for Testing in the Large

3.2 Solution: Testing in the Large
   3.2.1 Accepting the Master Test/QA Plan
   3.2.2 Recording Defects
   3.2.3 Regression Testing
   3.2.4 System Testing
   3.2.5 User Testing

3.3 Solution: Project Tasks
   3.3.1 Managing the Test in the Large Stage
   3.3.2 Training and Education
   3.3.3 People Management
   3.3.4 Quality Assurance
   3.3.5 Potential Risks While Testing in the Large
   3.3.6 Opportunities for Reuse
   3.3.7 Metrics

3.4 Resulting Context: Exit Conditions for Testing in the Large

3.5 Secrets of Success

3.6 Process Checklist

3.7 What You Have Learned in this Chapter

3.8 References and Recommended Reading

Chapter 4
The Rework Stage

4.1 Initial Context: Entry Conditions for Reworking Your Application

4.2 Solution: Reworking Your Application
   4.2.1 Prioritizing Defects
   4.2.2 Fixing Defects
Contents

4.3 Solution: Project Tasks 90
  4.3.1 Managing the Rework Stage 90
  4.3.2 People Management 90
  4.3.3 Quality Assurance 91
  4.3.4 Potential Risks While Reworking Your Application 91
  4.3.5 Opportunities for Reuse 92
  4.3.6 Metrics 92

4.4 Resulting Context: Exit Conditions for Reworking Your Application 93

4.5 Secrets of Success 93

4.6 Process Checklist 94

4.7 What You Have Learned in This Chapter 95

4.8 References and Recommended Reading 96

Chapter 5

The Release Stage 97

5.1 Initial Context: Entry Conditions for Releasing Your Application 98

5.2 Solution: Releasing Your Application 99
  5.2.1 Preparing for Release 100
  5.2.2 Releasing the Application to Operations and Support 108
  5.2.3 Releasing the Application to the User Community 112

5.3 Solution: Project Tasks 115
  5.3.1 Managing the Release Stage 116
  5.3.2 People Management 117
  5.3.3 Training and Education 118
  5.3.4 Quality Assurance 118
  5.3.5 Potential Risks While Releasing Your Application 119
  5.3.6 Opportunities for Reuse 119
  5.3.7 Metrics 120
Contents

iv

5.4 Resulting Context: Exit Conditions for Releasing Your Application 121
5.5 Secrets of Success 122
5.6 Process Checklist 122
5.7 What You Have Learned in This Chapter 124
5.8 References and Recommended Reading 124

Chapter 6

The Assess Stage 125

6.1 Initial Context: Entry Conditions for Assessing Your Efforts 126
6.2 Solution: Assessing Your Efforts 128
   6.2.1 Assessing the Project and the Project Team 130
   6.2.2 Documenting the Assessment 136
6.3 Solution: Project Tasks 138
   6.3.1 Managing the Assess Stage 138
   6.3.2 People Management 139
   6.3.3 Quality Assurance 139
   6.3.4 Potential Risks While Assessing Your Efforts 139
   6.3.5 Opportunities for Reuse 140
   6.3.6 Metrics 141
6.4 Resulting Context: Exit Conditions for Assessing Your Efforts 141
6.5 Secrets of Success 142
6.6 Process Checklist 143
6.7 What You Have Learned in This Chapter 144
6.8 References and Recommended Reading 144
### Contents

*Part 2  Maintain and Support 147*

**Chapter 7**

**The Maintain and Support Phase 149**

7.1  Initial Context: Entry Conditions to the Maintain and Support Phase 153

7.2  Solution: How Work Generally Proceeds During the Maintain and Support Phase 154

7.3  Solution: Project Tasks 155

7.3.1  Managing the Maintain and Support Phase 156

7.3.2  People Management 156

7.3.3  Potential Risks During the Maintain and Support Phase 156

7.3.4  Training and Education Issues 158

7.4  Resulting Context: Exit Conditions from the Maintain and Support Phase 158

7.5  Secrets of Success 159

7.6  Process Checklist 160

7.7  What You Have Learned in This Chapter 160

7.8  References and Recommended Reading 161

**Chapter 8**

**The Support Stage 163**

8.1  Initial Context: Entry Conditions for Supporting Your Application 164

8.2  Solution: Supporting Your Application 165

8.2.1  Responding to Support Requests 165

8.2.2  Determining a Resolution 166

8.2.3  Resolving the Issue 171
## Contents

*xvi*

8.3 Solution: Project Tasks 173  
8.3.1 Managing the Support Stage 173  
8.3.2 People Management 177  
8.3.3 Training and Education 178  
8.3.4 Quality Assurance 178  
8.3.4 Potential Risks While Supporting Your Application 179  
8.3.5 Opportunities for Reuse 179  
8.3.6 Metrics 180  

8.4 Resulting Context: Exit Conditions for Supporting Your Application 181  

8.5 Secrets of Success 181  

8.6 Process Checklist 182  

8.7 What You Have Learned in This Chapter 183  

8.8 References and Recommended Reading 183  

### Chapter 9  

The Identify Defects and Enhancements Stage 185  

9.1 Initial Context: Entry Conditions for Identification of Defects and Enhancements 186  

9.2 Solution: Identifying Defects and Enhancements 187  
9.2.1 Analyzing Software Change Requests (SCRs) 188  
9.2.2 Prioritizing Maintenance Changes 189  
9.2.3 Allocating Maintenance Changes to Configuration Items 190  

9.3 Solution: Project Tasks 192  
9.3.1 Managing the Identify Defects and Enhancements Stage 192  
9.3.2 People Management 192  
9.3.3 Training and Education 193  
9.3.4 Quality Assurance 193  
9.3.5 Potential Risks While Identifying Defects and Enhancements 193
## Contents

*xxvii*

9.3.6 Opportunities for Reuse 194
9.3.7 Metrics 194

9.4 Resulting Context: Exit Conditions for Identification of Defects and Enhancements 195

9.5 Secrets of Success 195

9.6 Process Checklist 196

9.7 What You Have Learned in This Chapter 197

9.8 References and Recommended Reading 197

### Part 3 Tying Up Loose Ends 199

**Chapter 10**

The Project and Cross-Project Tasks of the OOSP 201

10.1 Managing Projects 201
   10.1.1 The Mechanics of Project Management 203
   10.1.2 Milestones 204
   10.1.3 Secrets of Success 206

10.2 People Management 208
   10.2.1 Working Together 208
   10.2.2 Reward Systems 209
   10.2.3 Politics, Power Struggles, and Other Realities 210
   10.2.4 Staffing an Object-Oriented Project 212
   10.2.5 Secrets of Success 214

10.3 Training and Education 215
   10.3.1 The Expected Lifespan of Skills 216
   10.3.2 Getting People Started with Objects 217
   10.3.3 Successful Approaches to Classroom Training and Education 222
   10.3.4 Successful Approaches to Mentoring 223
   10.3.5 Secrets of Success 224
Contents

10.4 Quality Assurance 227
  10.4.1 Quality Assurance Throughout Your Project 228
  10.4.2 A Few Words About ISO 9000 230
  10.4.3 Secrets of Success 231

10.5 Risk Management 233
  10.5.1 The Sources of Risk 234
  10.5.2 The Risk Management Process Pattern 236
  10.5.3 Potential Risks Throughout Your Project 238
  10.5.4 Secrets of Success 238

10.6 Reuse Management 242
  10.6.1 Types of Reuse 244
  10.6.2 Reuse Throughout Your Project 252
  10.6.3 The Reuse Capability Model 252
  10.6.4 Reuse Tools 252
  10.6.5 Secrets of Success 257

10.7 Metrics Management 260
  10.7.1 Why Should You Collect Metrics? 260
  10.7.2 Collecting Metrics Throughout Your Project 261
  10.7.3 Metrics Categories 261
  10.7.4 Metrics Applicable to the Entire OOSP 261
  10.7.5 Establishing a Metrics Program 263
  10.7.6 Secrets of Success 266

10.8 Managing Deliverables 268
  10.8.1 Software Configuration Management 268
  10.8.2 The Qualities of Good Deliverables 268
  10.8.3 Deliverables Throughout Your Project 269
  10.8.4 Secrets of Success 272

10.9 Managing Infrastructure 273
  10.9.1 Infrastructure Throughout Your Project 274
  10.9.2 Secrets of Success 274

10.10 What You Have Learned in This Chapter 276

10.11 References and Recommended Reading 276
## Contents

**Chapter 11**

### Introducing the OOSP into Your Organization 281

11.1 The Capability Maturity Model (CMM) 283
   - 11.1.1 The Five CMM Maturity Levels 284
   - 11.1.2 Understanding the Key Process Areas (KPAs) 287
   - 11.1.3 Mapping the Key Process Areas to Our OOSP 287
   - 11.1.4 A Potential Schedule for Introducing Process Patterns Into Your Organization 287

11.2 Introducing Process Patterns Into Your Organization 287
   - 11.2.1 Initiate—Getting Your Process Improvement Program Started 291
   - 11.2.2 Diagnose—Assess Your Situation and Identify Your Goals 296
   - 11.2.3 Establish—Organizing Your Process Improvement Program 299
   - 11.2.4 Act—Defining Process Processing for Your Organization 305
   - 11.2.5 Act—Implementing Processes 311
   - 11.2.6 Leverage—Assess and Iterate 312

11.3 Rethinking Your Organizational Structure 314

11.4 Secrets of Success 319

11.5 What You Have Learned in This Chapter 325

11.6 References and Recommended Reading 325

**Glossary** 329

**Index** 361
Illustrations

**Figure 1.1.** The Object-Oriented Software Process (OOSP) 9

**Figure 1.2.** The Detailed Modeling process pattern 15

**Figure 1.3.** The aggressive approach to incremental development 21

**Figure 1.4.** A standard approach to incremental development 23

**Figure 1.5.** A conservative approach to incremental development 23

**Figure 1.6.** The five CMM maturity levels 32

**Figure 1.7.** The key process areas of the SEI CMM (SEI, 1995) 33

**Figure 2.1.** The Deliver process pattern 44

**Figure 2.2.** Organizing the Deliver phase 47

**Figure 2.3.** Organizing the work of the Deliver phase 48

**Figure 2.4.** Potential roles during the Deliver phase 51

**Figure 3.1.** The Test in the Large stage 60

**Figure 3.2.** The techniques of Full Life cycle Object-Oriented Testing (FLOOT) 61

**Figure 3.3.** The Test in the Large process pattern 63

**Figure 4.1.** The Rework stage 84
### Illustrations

<table>
<thead>
<tr>
<th>Illustration</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 4.2.</td>
<td>The Rework process pattern</td>
<td>86</td>
</tr>
<tr>
<td>Figure 5.1.</td>
<td>The Release stage</td>
<td>98</td>
</tr>
<tr>
<td>Figure 5.2.</td>
<td>The Release process pattern</td>
<td>101</td>
</tr>
<tr>
<td>Figure 6.1.</td>
<td>The Assess stage</td>
<td>126</td>
</tr>
<tr>
<td>Figure 6.2.</td>
<td>The Assess process pattern</td>
<td>129</td>
</tr>
<tr>
<td>Figure 7.1.</td>
<td>The Maintain and Support process pattern</td>
<td>150</td>
</tr>
<tr>
<td>Figure 7.2.</td>
<td>Comparing approaches to releasing software</td>
<td>152</td>
</tr>
<tr>
<td>Figure 7.3.</td>
<td>A conservative approach to incremental development</td>
<td>153</td>
</tr>
<tr>
<td>Figure 7.4.</td>
<td>Organizing the Maintain and Support phase</td>
<td>154</td>
</tr>
<tr>
<td>Figure 7.5.</td>
<td>Organizing the work of the Maintain and Support phase</td>
<td>155</td>
</tr>
<tr>
<td>Figure 7.6.</td>
<td>Potential roles during the Maintain and Support phase</td>
<td>156</td>
</tr>
<tr>
<td>Figure 8.1.</td>
<td>The Support stage</td>
<td>164</td>
</tr>
<tr>
<td>Figure 8.2.</td>
<td>The Support process pattern</td>
<td>165</td>
</tr>
<tr>
<td>Figure 9.1.</td>
<td>The Identify Defects and Enhancements stage</td>
<td>186</td>
</tr>
<tr>
<td>Figure 9.2.</td>
<td>The Identify Defects and Enhancements process pattern</td>
<td>187</td>
</tr>
<tr>
<td>Figure 10.1.</td>
<td>The Object-Oriented Software Process (OOSP)</td>
<td>202</td>
</tr>
<tr>
<td>Figure 10.2.</td>
<td>Relative staffing requirements, by project stage, for an object-oriented project</td>
<td>213</td>
</tr>
<tr>
<td>Figure 10.3.</td>
<td>Relative staffing levels during the phases of an object-oriented project</td>
<td>214</td>
</tr>
<tr>
<td>Figure 10.4.</td>
<td>Comparing the expected lifespan of skills</td>
<td>217</td>
</tr>
<tr>
<td>Figure 10.5.</td>
<td>Object-oriented training and education</td>
<td>218</td>
</tr>
<tr>
<td>Figure 10.6.</td>
<td>The Technical Review process pattern</td>
<td>231</td>
</tr>
<tr>
<td>Figure 10.7.</td>
<td>The Risk Management task process pattern</td>
<td>237</td>
</tr>
<tr>
<td>Figure 10.8.</td>
<td>The Develop Reusable Artifacts process pattern</td>
<td>243</td>
</tr>
<tr>
<td>Figure 10.9.</td>
<td>The Reuse First process pattern</td>
<td>244</td>
</tr>
<tr>
<td>Figure 10.10.</td>
<td>The effectiveness of the various approaches to reuse</td>
<td>245</td>
</tr>
<tr>
<td>Figure 10.11.</td>
<td>A component model for a telecommunications firm</td>
<td>251</td>
</tr>
<tr>
<td>Figure 10.12.</td>
<td>The Reuse Capability Model (RCM)</td>
<td>253</td>
</tr>
<tr>
<td>Figure 11.1.</td>
<td>The Object-Oriented Software Process (OOSP)</td>
<td>282</td>
</tr>
</tbody>
</table>
Illustrations

xxiii

Figure 11.2. The five CMM maturity levels 284
Figure 11.3. The Object-Oriented Software Process (OOSP) 307
Figure 11.4. The Release process pattern 309
Figure 11.5. A tailored version of the Release process pattern 310
Figure 11.6. The Object-Oriented Software Process (OOSP) 315
Figure 11.7. An organization structure 320
Figure 11.8. What really happened to the Neanderthals 322
Abbreviations

4GL    Fourth-generation language
ACD    Automatic call distribution
BDE    Business domain expert
C/S    Client/server
CASE   Computer-aided system engineering
CBT    Computer-based training
CCB    Configuration control board
CI     Configuration item
CM     Configuration management
CMM    Capability Maturity Model
CORBA  Common Object Request Broker Architecture
COTS   Commercial-off-the-shelf
CPU    Central processing unit
CR     Change request
Abbreviations
xxvi

CRC     Class responsibility collaborator
CRUD    Create, read, update, delete
DB      Database
DBA     Database administrator
DFD     Data-flow diagram
ER      Entity-relationship
FLOOT   Full life cycle object-oriented testing
GUI     Graphical user interface
HFE     Human factors engineer
S'MIDEAL\(^1\)    Initiating, Diagnosing, Establishing, Acting, and Leveraging
IRR     Internal rate of return
IS      Information system
ISO     International Standards Organization
IT      Information technology
JAD     Joint application development
JIT     Just in time
JRP     Joint requirements planning
KISS    Keep it simple, silly
KPA     Key process area
MIS     Management information system
NCSS    Non-comment source statement
NIH     Not invented here
OODBMS  Object-oriented database management system
OID     Object identifier
OMG     Object Management Group

\(^1\) S'MIDEAL is a service mark of Carnegie Mellon University.
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OML</td>
<td>OPEN modeling language</td>
</tr>
<tr>
<td>OO</td>
<td>Object-oriented</td>
</tr>
<tr>
<td>OO</td>
<td>Object-orientation</td>
</tr>
<tr>
<td>OOA</td>
<td>Object-oriented analysis</td>
</tr>
<tr>
<td>OOD</td>
<td>Object-oriented design</td>
</tr>
<tr>
<td>OOP</td>
<td>Object-oriented programming</td>
</tr>
<tr>
<td>OOSP</td>
<td>Object-Oriented Software Process</td>
</tr>
<tr>
<td>OOUI</td>
<td>Object-oriented user interface</td>
</tr>
<tr>
<td>OPEN</td>
<td>Object-Oriented, Process, Environment, and Notation</td>
</tr>
<tr>
<td>OTC</td>
<td>Object technology center</td>
</tr>
<tr>
<td>PAT</td>
<td>Process action team</td>
</tr>
<tr>
<td>PERT</td>
<td>Program Evaluation and Review Technique</td>
</tr>
<tr>
<td>PMI</td>
<td>Project Management Institute</td>
</tr>
<tr>
<td>QA</td>
<td>Quality assurance</td>
</tr>
<tr>
<td>RAD</td>
<td>Rapid application development</td>
</tr>
<tr>
<td>RAM</td>
<td>Requirements allocation matrix</td>
</tr>
<tr>
<td>RDB</td>
<td>Relational database</td>
</tr>
<tr>
<td>SCM</td>
<td>Software configuration management</td>
</tr>
<tr>
<td>SCR</td>
<td>Software change request</td>
</tr>
<tr>
<td>SCRB</td>
<td>Software configuration review board</td>
</tr>
<tr>
<td>SDLC</td>
<td>System development life cycle</td>
</tr>
<tr>
<td>SEI</td>
<td>Software Engineering Institute</td>
</tr>
<tr>
<td>SEPG</td>
<td>Software Engineering Process Group</td>
</tr>
<tr>
<td>SME</td>
<td>Subject-matter expert</td>
</tr>
<tr>
<td>SPICE</td>
<td>Software Process Improvement and Capability dEtermination</td>
</tr>
</tbody>
</table>
### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPR</td>
<td>Software problem report</td>
</tr>
<tr>
<td>SQA</td>
<td>Software quality assurance</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured query language</td>
</tr>
<tr>
<td>SRS</td>
<td>System requirement specification</td>
</tr>
<tr>
<td>T&amp;E</td>
<td>Training and education</td>
</tr>
<tr>
<td>UAT</td>
<td>User-acceptance testing</td>
</tr>
<tr>
<td>UI</td>
<td>User interface</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>V&amp;V</td>
<td>Verification &amp; Validation</td>
</tr>
<tr>
<td>VDD</td>
<td>Version description document</td>
</tr>
<tr>
<td>WBS</td>
<td>Work breakdown structure</td>
</tr>
<tr>
<td>Y2K</td>
<td>Year 2000</td>
</tr>
</tbody>
</table>
Foreword

This book continues a new approach to understanding the key development issues that constantly distract important software projects from delivering effective software, because they aren’t documented in most organizations. It has been known for many years that an effective software development lifecycle will deliver good software with fewer defects than the uncontrolled random iteration of code and requirement. The initial book defined the core development stages and lifecycle. The second volume provides the management, testing, and delivery phases of development.

The world of software and system development is changing. Tools and languages are starting to gain de facto standards that are allowing projects to develop the more complex systems being demanded by the business and end users. As more people join the software industry, experience is being stretched and it is becoming more difficult to find senior developers with more than three strong projects behind them. Communicating experience is becoming one of the most important success factors to most development teams. Quality systems are fine for defining the letter of the law and rules that must be obeyed to keep a project on track, but they often miss the guidance and explanation of why it
Foreword

xxx

is important to follow the steps outlined in their often hundreds of pages. These can make quality systems very unapproachable by the average developer.

Scott’s book provides an intuitive and clearly written guide to the key issues, concepts, deliverables and processes needed to deliver a successful project. He writes as if he is your own personal mentor, sitting next to you, answering the questions that face every project, team, manager, and developer.

The book uses the new and evolving form of patterns to structure and communicate a wide range of personal experience and best practice—the things those most seasoned system and software professionals all know, but progressively forget to communicate to the teams and developers they work with. “It’s common knowledge” is often the excuse, but the pace of software development and the constantly changing faces in many companies means that it becomes common to only a few. This book and its predecessor provides a wealth of real practical knowledge that every developer needs but often misses out on.

With this book Scott has set a foundation of a next generation of process and pattern development that will evolve, I believe, toward a standard way of describing, documenting, and structuring experience and knowledge about software and system development processes. There is still much work to be done in this arena, but this book (and its companion) will provide an invaluable reference of concepts and processes that will allow many projects to recognize and formalize their own development process.

I would expect that many IT professionals will recognize much of the advice within the book, but will gain considerably for being able to read and relate the knowledge with a development lifecycle. By following the patterns presented in the book and balancing the forces and drivers of their own project, they will improve not only their own personal performance and effectiveness, but will start to raise the process maturity of the team and project they apply the process patterns to.

Don Kavanagh,
Principal Methodologist, SSA Object Technology
Preface

Organizations have moved beyond the pilot project stage and are now using object technology to build large-scale, mission-critical business applications. Unfortunately they are finding that the processes that proved so successful on small, proof-of-concept projects do not scale very well for real-world development. Today’s organization needs a collection of proven techniques for managing the complexities of large-scale, object-oriented software development projects: a collection of process patterns. A process pattern describes a collection of general techniques, actions, and/or tasks for developing object-oriented software. In many ways process patterns are the reusable building blocks from which your organization can tailor a mature software process. To be fair, however, it isn’t enough to just have a collection of process patterns, you must have a pattern language of process patterns that fit together as a consistent whole, that form a complete software process. That is what Process Patterns and its sister book More Process Patterns provide.

The object-oriented software process (OOSP) presented in these two volumes is a collection of process patterns that are geared toward medium to large-size organizations that need to develop
Preface

xxxii

software to support their main line of business. Although the OOSP could easily be modified for the development of shrink-wrapped software, I would rather point you in the direction of the Unified software process. The Unified software process was created by a leading maker of shrink-wrapped development tools. There is no such thing as a one-size-fits-all process, and the OOSP is just one of several approaches to developing object-oriented (OO) software.

I have chosen to describe the OOSP as a collection of process patterns that have been proven in practice: they are not the theoretical musings of an ivory-tower academic who has never built software. The OOSP provides a framework which addresses issues such as how to:

• successfully deliver large applications using object technology
• develop applications that are truly easy to maintain and enhance
• manage these projects
• ensure that your development efforts are of high quality.

Dispelling Industry Myths

These two books actively attack several myths of the object industry. First, the belief that object development is a purely iterative process is questionable at best. Although this might appear to be true for small pilot projects using OO technology, the reality for large-scale, mission-critical applications is that the OO development process is serial in the large and iterative in the small, delivering incremental releases over time. At the time of this writing, the interesting thing is that the Unified software process is in fact based on this concept, although they still claim that they have an incremental process. Oh well; old myths die hard, I guess.

Second, these books also disprove the myth that you can do less testing when you use OO technology. The reality is that the exact opposite is true: you need to do more. One of my fundamental beliefs, a belief shared by the vast majority of professional
Preface

software engineers, is that testing and quality assurance should be performed throughout the entire development process, not just at the end of it. Furthermore, the reality of incremental development is that you need to perform more regression testing than with single-shot, "big-bang" development.

Third, these two books disprove the myth that we only need to be concerned with development issues while building an application. The reality is that the concerns of maintenance and support are just as important, if not more so, than those of development. The OOSP explicitly includes both maintenance and support as part of the project life cycle, so as to put it in the face of everyone involved in the development process. There is a saying in the computer industry: A good developer knows that there is more to development than programming; a great developer knows that there is more to development than development.

Fourth, I hope that these books disprove the myth that processes only result in needless paperwork. My experience has been that process patterns, when applied intelligently, increase the productivity of developers. My experience has also been that when process patterns are applied less than intelligently — that when the paper pushers have too much influence in an organization — they can decrease your productivity. The process patterns of the OOSP pattern language have been proven to work in practice — how you choose to implement them will determine how successful you are. Organizations that keep the end goal in mind — that of developing, maintaining, and supporting software that fulfills the needs of their user community — will be successful with process patterns. Those that follow processes simply for the sake of following processes are likely to fail.

The Object-Oriented Software Process (OOSP)

So what do these books cover? First of all, they are organized by the four serial phases of OO development: Initiate, Construct, Deliver, and Maintain and Support. Second, each phase is then divided further into its iterative stages. As a result, the two books are organized into the following chapters: