SOA Design Patterns

SOA Design Patterns

Thomas Erl

(with additional contributors)

PRENTICE HALL

UPPER SADDLE RIVER, NJ • BOSTON • INDIANAPOLIS • SAN FRANCISCO

NEW YORK • TORONTO • MONTREAL • LONDON • MUNICH • PARIS • MADRID

CAPETOWN • SYDNEY • TOKYO • SINGAPORE • MEXICO CITY

Library of Congress Cataloging-in-Publication Data:

Erl, Thomas.

SOA design patterns / Thomas Erl. — 1st ed.

p. cm.

ISBN 0-13-613516-1 (hardback : alk. paper) 1. Web services. 2. Computer architecture. 3. Software patterns. 4. System design. I. Title.

TK5105.88813.E735 2008

006.7—dc22

2008040488

ISBN-13: 978-0-13-613516-6
ISBN-10: 0-13-613516-1
Text printed in the United States on recycled paper at R.R. Donnelley in Crawfordsville, Indiana.
First printing December 2008

The following patterns: Exception Shielding, Threat Screening, Trusted Subsystem, Service Perimeter Guard, Data Confidentiality, Data Origin Authentication, Direct Authentication, Brokered Authentication are courtesy of the Microsoft Patterns & Practices team. For more information please visit http://msdn.microsoft.com/practices. These patterns were originally developed by Jason Hogg, Frederick Chong, Dwayne Taylor, Lonnie Wall, Paul Slater, Tom Hollander, Wojtek Kozaczynski, Don Smith, Larry Brader, Sajjas Nasir Imran, Pablo Cibraro, Nelly Delgado and Ward Cunningham

Editor-in-Chief
Mark L. Taub

Managing Editor
Kristy Hart

Copy Editor
Language Logistics

Indexer
Cheryl Lenser

Proofreader
Williams Woods Publishing

Composition
Jake McFarland
Bumpy Design

Graphics
Zuzana Cappova
Tami Young
Spencer Fruhling

Photos
Thomas Erl

Cover Design
Thomas Erl

To the SOA pioneers that blazed the trail we now so freely base our roadmaps on, and to the SOA community that helped me refine the wisdom of the pioneers into this catalog of patterns.

- Thomas Erl

Contents

Foreword xxxvii

CHAPTER 1: Introduction 1

1.1 Objectives of this Book 4
1.2 Who this Book is For 4
1.3 What this Book Does Not Cover 4
Topics Covered by Other Books 4
Web Service and REST Service Design Patterns 5
SOA Standardization Efforts 5
1.4 Recommended Reading 6
1.5 How this Book is Organized 7
Part I: Fundamentals 8
Part II: Service Inventory Design Patterns 8
Part III: Service Design Patterns 8
Part IV: Service Composition Design Patterns 9
Part V: Supplemental 10
Part VI: Appendices 10
1.6 Symbols, Figures, Style Conventions 11
Symbol Legend 11
How Color is Used 11
Data Flow and Directionality Conventions. 11
Pattern Documentation Conventions. 11
1.7 Additional Information 11
Updates, Errata, and Resources (www.soabooks.com) 11
Visio Stencil (www.soabooks.com) 12

Community Patterns Site (www.soapatterns.org) 12
Master Glossary (www.soaglossary.com) 12
Supplementary Posters (www.soaposters.com) 12
The SOA Magazine (www.soamag.com) 12
Referenced Specifications (www.soaspecs.com). 12
Notification Service 13
Contact the Author 13

Chapter 2: Case Study Background 15

2.1 Case #1 Background: Cutit Saws Ltd. 17
History 18
Technical Infrastructure and Automation Environment 18
Business Goals and Obstacles. 18

2.2 Case #2 Background: Alleywood Lumber Company 19
History 19
Technical Infrastructure and Automation Environment 20
Business Goals and Obstacles. 20

2.3 Case #3 Background: Forestry Regulatory Commission (FRC) 21
History 21
Technical Infrastructure and Automation Environment 21
Business Goals and Obstacles. 22

PART I: FUNDAMENTALS

Chapter 3: Basic Terms and Concepts 25

Purpose of this Introductory Chapter 26

3.1 Architecture Fundamentals 26
A Classic Analogy for Architecture and Infrastructure 27
Technology Architecture 27
Technology Infrastructure 30
Software Program 32
Relationship to Design Framework 33

3.2 Service-Oriented Computing Fundamentals. 35
Service-Oriented Computing 35
Service-Orientation 36
Service-Oriented Architecture (SOA) 37
Service 37
Service Capability 38
Service Consumer 38
Service Composition 40
Service Inventory. 42
Service-Oriented Analysis. 43
Service Candidate. 44
3.3 Service Implementation Mediums. 44
Services as Components 45
Services as Web Services 45
REST Services. 46

Chapter 4: The Architecture of Service-Orientation . . 47

Purpose of this Introductory Chapter 48
4.1 The Method of Service-Orientation 48
Principles of Service-Orientation. 48
Strategic Goals of Service-Oriented Computing. 51
4.2 The Four Characteristics of SOA. 52
Business-Driven 53
Vendor-Neutral 54
Enterprise-Centric 58
Composition-Centric 59
4.3 The Four Common Types of SOA 61
Service Architecture 62
Information Hiding 64
Design Standards. 64
Service Contracts 65
Service Agents 67
Service Capabilities 68
Service Composition Architecture 68
Nested Compositions 72
Task Services and Alternative Compositions 73
Compositions and Infrastructure. 74

Service Inventory Architecture 74
Service-Oriented Enterprise Architecture 76
Architecture Types and Scope 77
Architecture Types and Inheritance 77
Other Forms of Service-Oriented Architecture 78
Inter-Business Service Architecture 78
Service-Oriented Community Architecture 78

4.4 The End Result of Service-Orientation 79

CHAPTER 5: Understanding SOA Design Patterns 85

Purpose of this Introductory Chapter 86

5.1 Fundamental Terminology 86
What's a Design Pattern? 86
What's a Compound Pattern? 88
What's a Design Pattern Language? 88
What's a Design Pattern Catalog? 89

5.2 Historical Influences 89
Alexander's Pattern Language 90
Object-Oriented Patterns 91
Software Architecture Patterns 92
Enterprise Application Architecture Patterns 93
EAI Patterns 93
SOA Patterns 94

5.3 Pattern Notation 95
Pattern Symbols 95
Pattern Figures 96
Pattern Application Sequence Figures 96
Pattern Relationship Figures 96
Compound Pattern Hierarchy Figures 99
Capitalization 100
Page Number References 100

5.4 Pattern Profiles 100
Requirement 101
Icon 101
Summary 102
Problem 102
Solution 102

Application 103
Impacts 103
Relationships 103
Case Study Example 103

5.5 Patterns with Common Characteristics 104
Canonical Patterns 104
Centralization Patterns 105

5.6 Key Design Considerations 106
“Enterprise” vs. “Enterprise-wide” 106
Design Patterns and Design Principles 106
Design Patterns and Design Granularity 107
Measures of Design Pattern Application 108

PART II: SERVICE INVENTORY DESIGN PATTERNS

CHAPTER 6: Foundational Inventory Patterns 111

How Inventory Design Patterns Relate to SOA Design Characteristics 113
How Foundational Inventory and Service Patterns Relate 114
How Case Studies are Used in this Chapter 114

6.1 Inventory Boundary Patterns 114

Enterprise Inventory 116
Problem 116
Solution 117
Application 118
Impacts 120
Relationships 121
Case Study Example 122

Domain Inventory 123
Problem 123
Solution 124
Application 125
Impacts 126
Relationships 127
Case Study Example 128

6.2 Inventory Structure Patterns 130

Service Normalization. 131

Problem 131

Solution 132

Application. 132

Impacts 133

Relationships. 133

Case Study Example. 135

Logic Centralization 136

Problem 136

Solution 137

Application. 137

Impacts 139

Relationships. 140

Case Study Example. 142

Service Layers 143

Problem 143

Solution 144

Application. 145

Impacts 147

Relationships. 147

Case Study Example. 148

6.3 Inventory Standardization Patterns 149

Canonical Protocol 150

Problem 151

Solution 152

Application. 153

Impacts 155

Relationships. 155

Case Study Example. 157

Canonical Schema 158

Problem 158

Solution 159

Application. 159

Impacts 159

Relationships. 160

Case Study Example. 161

Chapter 7: Logical Inventory Layer Patterns 163

Combining Layers 164
Business Logic and Utility Logic 166
Agnostic Logic and Non-Agnostic Logic 166
Service Layers and Logic Types 167

Utility Abstraction 168
Problem 168
Solution 169
Application 170
Impacts 171
Relationships 171
Case Study Example 173

Entity Abstraction 175
Problem 175
Solution 176
Application 176
Impacts 178
Relationships 178
Case Study Example 180

Process Abstraction 182
Problem 182
Solution 183
Application 184
Impacts 185
Relationships 185
Case Study Example 187

Chapter 8: Inventory Centralization Patterns 191

Process Centralization 193
Problem 193
Solution 194
Application 195
Impacts 196
Relationships 197
Case Study Example 198

000_0136135161_FM.qxd 11/24/08 4:43 PM Page xx

Schema Centralization 200
Problem 200
Solution 201
Application 202
Impacts 202
Relationships 203
Case Study Example 203

Policy Centralization 207
Problems 207
Solution 208
Application 209
Impacts 210
Relationships 211
Case Study Example 213

Rules Centralization 216
Problem 216
Solution 217
Application 217
Impacts 218
Relationships 219
Case Study Example 222

CHAPTER 9: Inventory Implementation Patterns 225

Dual Protocols 227
Problem 228
Solution 228
Application 228
Impacts 233
Relationships 234
Case Study Example 235

Canonical Resources 237
Problem 238
Solution 238
Application 239
Impacts 239

000_0136135161_FM.qxd 11/24/08 4:43 PM Page xxi

Relationships 239
Case Study Example 241

State Repository 242
Problem 242
Solution 243
Application 244
Impacts 244
Relationships 244
Case Study Example 246

Stateful Services 248
Problem 248
Solution 248
Application 250
Impacts 250
Relationships 250
Case Study Example 251

Service Grid 254
Problem 254
Solution 255
Application 256
Impacts 257
Relationships 258
Case Study Example 259

Inventory Endpoint 260
Problem 260
Solution 261
Application 262
Impacts 263
Relationships 263
Case Study Example 265

Cross-Domain Utility Layer 267
Problem 267
Solution 268
Application 269
Impacts 269
Relationships 270
Case Study Example 270

000_0136135161_FM.qxd 11/24/08 4:43 PM Page xxii

CHAPTER 10: Inventory Governance Patterns 273

Canonical Expression. 275

Problem 275
Solution 275
Application. 276
Impacts 277
Relationships. 278
Case Study Example. 279

Metadata Centralization 280

Problem 280
Solution 281
Application. 282
Impacts 283
Relationships. 283
Case Study Example. 284

Canonical Versioning 286

Problem 286
Solution 287
Application. 287
Impacts 288
Relationships. 288
Case Study Example. 290

PART III: SERVICE DESIGN PATTERNS

CHAPTER 11: Foundational Service Patterns. 295

Case Study Background. 297
11.1 Service Identification Patterns 299

Functional Decomposition 300

Problem 300
Solution 301
Application. 302
Impacts 302
Relationships. 303
Case Study Example. 303

Service Encapsulation 305
Problem 305
Solution 306
Application 307
Impacts 309
Relationships 309
Case Study Example 310

11.2 Service Definition Patterns 311

Agnostic Context 312
Problem 313
Solution 314
Application 315
Impacts 315
Relationships 316
Case Study Example 317

Non-Agnostic Context 319
Problem 319
Solution 320
Application 321
Impacts 322
Relationships 322
Case Study Example 323

Agnostic Capability 324
Problem 324
Solution 325
Application 326
Impacts 327
Relationships 327
Case Study Example 328

CHAPTER 12: Service Implementation Patterns 331

Service Façade 333
Problem 333
Solution 334
Application 335

Impacts . . . 341
Relationships . . . 342
Case Study Example . . . 343

Redundant Implementation . . . 345
Problem . . . 345
Solution . . . 346
Application . . . 346
Impacts . . . 347
Relationships . . . 348
Case Study Example . . . 349

Service Data Replication . . . 350
Problem . . . 350
Solution . . . 352
Application . . . 353
Impacts . . . 353
Relationships . . . 353
Case Study Example . . . 354

Partial State Deferral . . . 356
Problem . . . 356
Solution . . . 357
Application . . . 358
Impacts . . . 359
Relationships . . . 359
Case Study Example . . . 360

Partial Validation . . . 362
Problem . . . 362
Solution . . . 363
Application . . . 364
Impacts . . . 364
Relationships . . . 364
Case Study Example . . . 365

UI Mediator . . . 366
Problem . . . 366
Solution . . . 367
Application . . . 368

Impacts 369
Relationships 370
Case Study Example 370

CHAPTER 13: Service Security Patterns 373

Case Study background 374

Exception Shielding 376
Problem 376
Solution 377
Application 378
Impacts 379
Relationships 379
Case Study Example 380

Message Screening 381
Problem 381
Solution 382
Application 382
Impacts 384
Relationships 385
Case Study Example 385

Trusted Subsystem 387
Problem 387
Solution 388
Application 388
Impacts 391
Relationships 391
Case Study Example 392

Service Perimeter Guard 394
Problem 394
Solution 395
Application 395
Impacts 396
Relationships 396
Case Study Example 397

CHAPTER 14: Service Contract Design Patterns 399

Decoupled Contract 401
Problem 401
Solution 402
Application 403
Impacts 405
Relationships 405
Case Study Example 407

Contract Centralization 409
Problem 409
Solution 410
Application 410
Impacts 411
Relationships 411
Case Study Example 413

Contract Denormalization 414
Problem 414
Solution 415
Application 416
Impacts 417
Relationships 417
Case Study Example 418

Concurrent Contracts 421
Problem 421
Solution 422
Application 423
Impacts 425
Relationships 425
Case Study Example 426

Validation Abstraction 429
Problem 429
Solution 430
Application 431
Impacts 432
Relationships 432
Case Study Example 433

Chapter 15: Legacy Encapsulation Patterns 439

Legacy Wrapper. 441
Problem 441
Solution 442
Application. 443
Impacts 444
Relationships. 444
Case Study Example. 446

Multi-Channel Endpoint 451
Problem 451
Solution 452
Application. 453
Impacts 454
Relationships. 454
Case Study Example. 456

File Gateway 457
Problem 457
Solution 458
Application. 458
Impacts 459
Relationships. 460
Case Study Example. 461

CHAPTER 16: Service Governance Patterns. 463

Compatible Change. 465
Problem 465
Solution 466
Application. 466
Impacts 469
Relationships. 469
Case Study Example. 470

Version Identification 472
Problem 472
Solution 473
Application. 473

Impacts 474
Relationships. 474
Case Study Example. 475

Termination Notification 478
Problem 478
Solution 479
Application. 480
Impacts 480
Relationships. 481
Case Study Example. 481

Service Refactoring. 484
Problem 484
Solution 485
Application. 485
Impacts 486
Relationships. 486
Case Study Example. 488

Service Decomposition. 489
Problem 489
Solution 491
Application. 492
Impacts 492
Relationships. 494
Case Study Example. 495

Proxy Capability 497
Problem 497
Solution 498
Application. 498
Impacts 500
Relationships. 500
Case Study Example. 501

Decomposed Capability 504
Problem 504
Solution 506
Application. 507
Impacts 507

Relationships 508
Case Study Example 508

Distributed Capability 510
Problem 510
Solution 511
Application 512
Impacts 513
Relationships 513
Case Study Example 514

PART IV: SERVICE COMPOSITION DESIGN PATTERNS

CHAPTER 17: Capability Composition Patterns 519

Capability Composition 521
Problem 521
Solution 521
Application 523
Impacts 523
Relationships 523
Case Study Example 524

Capability Recomposition 526
Problem 526
Solution 527
Application 527
Impacts 527
Relationships 529
Case Study Example 530

CHAPTER 18: Service Messaging Patterns 531

Service Messaging 533
Problem 533
Solution 533
Application 534
Impacts 534

Relationships 535
Case Study Example 536

Messaging Metadata 538
Problem 538
Solution 538
Application 539
Impacts 540
Relationships 541
Case Study Example 542

Service Agent 543
Problem 543
Solution 544
Application 544
Impacts 546
Relationships 546
Case Study Example 548

Intermediate Routing 549
Problem 549
Solution 551
Application 552
Impacts 553
Relationships 553
Case Study Example 556

State Messaging 557
Problem 557
Solution 558
Application 560
Impacts 561
Relationships 561
Case Study Example 562

Service Callback 566
Problem 566
Solution 568
Application 568
Impacts 570

000_0136135161_FM.qxd 11/24/08 4:43 PM Page xxxi

Relationships 570
Case Study Example 571

Service Instance Routing 574
Problem 574
Solution 576
Application 576
Impacts 578
Relationships 578
Case Study Example 579

Asynchronous Queuing 582
Problem 582
Solution 584
Application 584
Impacts 587
Relationships 588
Case Study Example 589

Reliable Messaging 592
Problem 592
Solution 593
Application 593
Impacts 594
Relationships 595
Case Study Example 596

Event-Driven Messaging 599
Problem 599
Solution 600
Application 602
Impacts 602
Relationships 602
Case Study Example 604

CHAPTER 19: Composition Implementation Patterns . . 605

Agnostic Sub-Controller 607
Problem 607
Solution 608

Application . . . 610
Impacts . . . 610
Relationships . . . 610
Case Study Example . . . 612

Composition Autonomy . . . 616
Problem . . . 616
Solution . . . 618
Application . . . 619
Impacts . . . 619
Relationships . . . 620
Case Study Example . . . 620

Atomic Service Transaction . . . 623
Problem . . . 623
Solution . . . 624
Application . . . 626
Impacts . . . 626
Relationships . . . 628
Case Study Example . . . 629

Compensating Service Transaction . . . 631
Problem . . . 631
Solution . . . 633
Application . . . 633
Impacts . . . 635
Relationships . . . 635
Case Study Example . . . 636

CHAPTER 20: Service Interaction Security Patterns . . 639

Data Confidentiality . . . 641
Problem . . . 641
Solution . . . 643
Application . . . 643
Impacts . . . 644
Relationships . . . 645
Case Study Example . . . 646

Data Origin Authentication 649
Problem 649
Solution 650
Application 651
Impacts 652
Relationships 653
Case Study Example 653

Direct Authentication 656
Problem 656
Solution 657
Application 657
Impacts 658
Relationships 659
Case Study Example 660

Brokered Authentication 661
Problem 661
Solution 662
Application 663
Impacts 665
Relationships 665
Case Study Example 666

CHAPTER 21: Transformation Patterns 669

Data Model Transformation 671
Problem 671
Solution 672
Application 673
Impacts 674
Relationships 674
Case Study Example 677

Data Format Transformation 681
Problem 681
Solution 681
Application 683

Impacts 683
Relationships. 683
Case Study Example. 685

Protocol Bridging 687
Problem. 687
Solution 688
Application. 688
Impacts 690
Relationships. 690
Case Study Example. 692

PART V: SUPPLEMENTAL

CHAPTER 22: Common Compound Design Patterns . . . 697

"Compound" vs. "Composite". 698
Compound Patterns and Pattern Relationships 698
Joint Application vs. Coexistent Application. 699
Compound Patterns and Pattern Granularity 700
Orchestration. 701
Enterprise Service Bus. 704
Service Broker. 707
Canonical Schema Bus. 709
Official Endpoint 711
Federated Endpoint Layer 713
Three-Layer Inventory. 715

CHAPTER 23: Strategic Architecture Considerations. . 717

Increased Federation 718
Increased Intrinsic Interoperability 721
Increased Vendor Diversification Options. 723

Increased Business and Technology Alignment 725
Increased ROI 727
Increased Organizational Agility 728
Reduced IT Burden 729

Chapter 24: Principles and Patterns at the U.S. Department of Defense 731

The Business Operating Environment (BOE) 733
Principles, Patterns, and the BOE 734
Incorporation of Information Assurance (IA) 736
Adherence to Standards 736
Data Visibility, Accessibility, and Understandability to Support Decision Makers 736
Loosely Coupled Services 736
Authoritative Sources of Trusted Data 737
Metadata-Driven Framework for Separation from Technical Details 737
Support Use of Open Source Software 738
Emphasize Use of Service-Enabled Commercial Off-the-Shelf (COTS) Software 738
Participation in the DoD Enterprise 738
Support Mobility — Users & Devices 738
The Future of SOA and the DoD 739
SOADoD.org 739

PART VI: APPENDICES

Appendix A: Case Study Conclusion 743

Cutit Saws Ltd. 744
Alleywood Lumber Company 744
Forestry Regulatory Commission (FRC) 745

Appendix B: Candidate Patterns 747

Appendix C: Principles of Service-Orientation 749

Standardized Service Contract 751
Service Loose Coupling 753
Service Abstraction 755
Service Reusability 756
Service Autonomy 758
Service Statelessness 760
Service Discoverability 762
Service Composability 764

Appendix D: Patterns and Principles Cross-Reference 767

Appendix E: Patterns and Architecture Types Cross-Reference 775

About the Author 783

About the Contributors 784

Index of Patterns 791

Index 795

Foreword

The entire history of software engineering can be characterized as one of rising levels of abstraction. We see this in our languages, our tools, our platforms, and our methods. Indeed, abstraction is the primary way that we as humans attend to complexity—and software-intensive systems are among the most complex artifacts ever created.

I would also observe that one of the most important advances in software engineering over the past two decades has been the practice of patterns. Patterns are yet another example of this rise in abstraction: A pattern specifies a common solution to a common problem in the form of a society of components that collaborate with one another. Influenced by the writings of Christopher Alexander, Kent Beck and Ward Cunningham began to codify various design patterns from their experience with Smalltalk. Growing slowly but steadily, these concepts began to gain traction among other developers. The publication of the seminal book *Design Patterns* by Erich Gamma, John Vlissides, Ralph Johnson, and Richard Helm marked the introduction of these ideas to the mainstream. The subsequent activities of the Hillside Group provided a forum for this growing community, yielding a very vibrant literature and practice. Now the practice of patterns is very much mainstream: Every well-structured software-intensive system tends to be full of patterns (whether their architects name them intentionally or not).

The emerging dominant architectural style for many enterprise systems is that of a service-oriented architecture, a style that at its core is essentially a message passing architecture. However, therein are many patterns that work (and anti-patterns that should be avoided).

Thomas' work is therefore the right book at the right time. He really groks the nature of SOA systems: There are many hard design decisions to be made, ranging from data-orientation to the problems of legacy integration and even security. Thomas offers wise counsel on each of these issues and many more, all in the language of design patterns. There are many things I like about this work. It's comprehensive. It's written in a very accessible pattern language. It offers patterns that play well with one another. Finally, Thomas covers not just the technical details, but also sets these patterns in the context of economic and other considerations.

SOA Design Patterns is an important contribution to the literature and practice of building and delivering quality software-intensive systems.

—*Grady Booch, IBM Fellow*
September, 2008

Acknowledgments

This book was in development for over three years, a good portion of which was dedicated to external reviews. Patterns were subjected to three review cycles that spanned a period of over twelve months and involved over 200 IT professionals. Pre-release galleys of my first and second manuscript drafts were printed and shipped to SOA experts and patterns experts around the world. Additionally, I had the full manuscript published at SOAPatterns.org for an open industry review. Even though these review phases added much time and effort to the development of this book, they ultimately elevated the quality of this work by a significant margin.

Special thanks to Prentice Hall for their patience and support throughout the book development process. Specifically, I'd like to thank Kristy Hart and Jake McFarland for their tremendous production efforts and tireless commitment to achieving printed perfection, Mark Taub who stood by this book project through a whirlwind of changes, reviews, more changes, extensions, and delays, Stephane Nakib and Heather Fox for their on-going guidance, and Eric Miller for his assistance with publishing the online review version of the first manuscript draft. I am fortunate to be working with the best publishing team in the industry.

Special thanks also to Herbjörn Wilhelmsen, Martin Fowler, Ralph Johnson, Bobby Woolf, Grady Booch, Gregor Hohpe, Baptist Eggen, Dragos Manolescu, Frank Buschmann, Wendell Ocasio, and Kevin Davis for their guidance and uninhibited feedback throughout the review cycles.

My thanks and gratitude to the following reviewers that participated in one or more of the manuscript reviews (in alphabetical order by last name):

Mohamad Afshar, Oracle

Sanjay Agara, Wipro

Stephen Bennett, Oracle

Steve Birkel, Intel

Brandon Bohling, Intel

Grady Booch, IBM

Bryan Brew, Booz Allen Hamilton

Victor Brown, CMGC

Frank Buschmann, Siemens

Enrique G. Castro-Leon, Intel

Peter Chang, Lawrence Technical University

Jason “AJ” Comfort, Booz Allen Hamilton

John Crupi, JackBe

Veronica Gacitua Decar, Dublin City University

Ed Dodds, Conmergence

Kevin P. Davis, PhD

Dominic Duggan, Stevens Institute of Technology

Baptist Eggen, Dutch Department of Defense

Steve Elston, Microsoft

Dale Ferrario, Sun Microsystems

Martin Fowler, ThoughtWorks

Pierre Fricke, Red Hat

Chuck Georgo, Public Safety and National Security

Larry Gloss, Information Manufacturing

Al Gough, CACI International Inc.

Daniel Gross, University of Toronto

Robert John Hathaway III, SOA Object Systems

William M. Hegarty, ThoughtWorks

Gregor Hohpe, Google

Ralph Johnson, UIUC

James Kinneavy, University of California

Robert Laird, IBM

Doug Lea, Oswego State University of New York

Canyang Kevin Liu, SAP

Terry Lottes, Northrop Grumman Mission Systems

Chris Madrid, Microsoft

Anne Thomas Manes, Burton Group

Dragos Manolescu, Microsoft

Steven Martin, Microsoft

Joe McKendrick

J.D. Meier, Microsoft

David Michalowicz, MITRE Corporation

Per Vonge Nielsen, Microsoft

Wendell Ocasio, DoD Military Health Systems, Agilex Technologies

Philipp Offermann, University of Berlin

Dmitri Ossipov, Microsoft

Prasen Palvakar, Oracle

Parviz Peiravi, Intel

Nishit Rao, Oracle

Ian Robinson, ThoughtWorks

Richard Van Schelven, Ericsson

Shakti Sharma, Sysco Corp

Don Smith, Microsoft

Michael Sor, Booz Allen Hamilton

John Sparks, Western Southern Life

Sona Srinivasan, CISCO

Linda Terlouw, Ordina

Phil Thomas, IBM

Steve Vinoski, IEEE

Herbjörn Wilhelmsen, Objectware

Peter B. Woodhull, Modus21

Bobby Woolf, IBM

Farzin Yashar, IBM

Markus Zirn, Oracle

Olaf Zimmermann, IBM

There were many more individuals who directly or indirectly supported this effort. Amidst the flurry of correspondence over the past three years, I was unable to keep track of all participants. If you were part of the SOA design patterns project and you don't see your name on this list, then do contact me via www.thomaserl.com.

Contributors

In alphabetical order by last name:

Larry Brader

David Chappell, Oracle

Frederick Chong

Pablo Cibraro, Lagash Systems SA

Ward Cunningham

Nelly Delgado, Microsoft

Florent Georges

Charles Stacy Harris, Microsoft

Kelvin Henney, Curbralan

Jason Hogg, Microsoft

Tom Hollander

Anish Karmarkar, Oracle

Sajjas Nasir Imran, Infosys

Berthold Maier, Oracle

Hajo Normann, EDS

Wojtek Kozaczynski

Mark Little, Red Hat

Brian Lokhorst, Dutch Tax Office

Brian Loesgen, Neudesic

Matt Long, Microsoft

David Orchard, Oracle

Thomas Rischbeck, IPT

Chris Riley, SOA Systems

Satadru Roy, Sun Microsystems

Arnaud Simon, Red Hat

Paul Slater, Wadeware

Don Smith

Sharon Smith, Microsoft

Dwayne Taylor

Tina Tech

Bernd Trops, SOPERA GmbH

Clemens Utschig-Utschig, Oracle

Lonnie Wall, RDA Corporation

Torsten Winterberg, Oracle

Dennis Wisnosky, U.S. Department of Defense

001_0136135161_01.qxd 11/24/08 6:41 PM Page 1

Chapter 1

Introduction

1.1 Objectives of this Book

1.2 Who this Book is For

1.3 What this Book Does Not Cover

1.4 Recommended Reading

1.5 How this Book is Organized

1.6 Symbols, Figures, Style Conventions

1.7 Additional Information

While recently delivering a week-long workshop at a client location, I was required to spend a fair amount of time waiting in the reception area every day. The client was a very large company in the finance industry, and in order to enter their offices, I had to first request a visitor's pass from the security guard and then wait until someone from the office came down to the lobby to escort me back up.

Upon entering the building for the first time, I noticed that the front door was stuck. It took me two or three tries to force it open. The security guard later told me that a delivery person had accidentally struck the door with some sort of cart, warping the frame and damaging the handle. They weren't expecting replacement parts to be installed for another two weeks and were not allowed to keep the door open.

While waiting for my escort that day, I noticed numerous people (mostly office staff) trying to access the building via the jammed door, each going through the same experience I did. People tried different approaches, some more effective than others. At one point there was an actual line-up impatiently waiting for the person at the front to figure it out. Just about everyone who eventually entered complained about the door to the guard.

On the next day of the workshop I was again waiting in the lobby watching the same story unfold. I saw familiar faces struggling with the door again; getting it ajar seemed more a matter of luck than technique, so it was difficult to remember how one opened it the day before. On this day, the security guard ran toward the door to help people open it whenever he could. However, over time, he found himself rushing back and forth a lot, dealing with the door and tending to people at the reception desk who needed to register and request passes.

The third day came around, and I was surprised to encounter the guard standing outside by the entrance. I could see through the glass walls that someone else was taking care of reception duties. As I approached the door, the guard greeted me, and I assumed he was going to open it to let me in. Instead, he asked me not to enter and proceeded to give me a short lesson on how to open the door with two swift moves. The lesson consisted of a brief explanation and a short demonstration. I thanked him and moved to go inside, but he stopped me, shut the door, and then said "Ok, now you do it." And so I did.

While waiting in the lobby that morning, I watched him give that same lesson to just about everyone who needed to enter. Sometimes the guard had a whole class as a group of office employees who arrived at the same time were taught together.

As I walked toward the building on the fourth day, I noticed the guard was no longer at the door. Recalling "the moves" I'd learned the day before, I proceeded to open it with relative ease. As I entered the lobby, I could see that the same guard was alone again to manage the reception area. Then, while waiting for my escort as usual, I witnessed droves of people entering the building with little to no problems.

It was even more impressive the following day during my last morning. People were coming and going without breaking a stride. It was as if the door had actually been fixed. At the end of that last day of training, I said good-bye to the guard and complimented him on how he dealt with the door issue. "You're a true problem solver," I said. "Yes, I know," he responded with a grin, "I'm the smart one."

On the taxi ride back to the airport, I thought some more about the damaged door and the solution the guard came up with. I did some rough math, taking into account how long it took the average person to get past the jammed door during the first two days and how many people I saw streaming into the building every morning. I estimated that over the last two days (after each employee was given a lesson by the guard) about 35,000 seconds were saved, translating into around 9.7 hours.

I never did find out when that door was eventually fixed, but assuming it took another week as expected, that time savings could easily be doubled or tripled. That's potentially 20–30 extra working hours the company received, thanks to one person's ingenuity.

This experience reminded me of why I felt strongly about putting together this catalog of design patterns. That guard spent the second day trying a variety of ways to deal with a problem until he found a proven method that was effective, easy to learn, and repeatable by anyone. On the third day he transferred that knowledge to all who needed it, and on the remaining days they put that knowledge to good use. In the end, the cumulative benefit was significant because all of the employees who saved time were able to spend that time solving new problems for the benefit of the company.

While problem solving is a fundamental skill we all possess, not everyone should have to solve the same problems. This is the basic rationale behind design patterns. There are jammed doors along the path to completing just about any IT project, perhaps even more so with SOA initiatives simply because their scope tends to be larger and more ambitious. I hope you'll find this book an effective resource for "learning the moves" to counter problems you might have to face in pursuit of realizing your own service-oriented solutions.

1.1 Objectives of this Book

A design pattern is simply a proven design solution for a common design problem that is formally documented in a consistent manner. This book was written with one primary goal in mind: to provide a master pattern catalog and pattern language for SOA and service-orientation. This sole objective has driven this collection of design patterns through numerous rounds of reviews, revisions, and community participation.

1.2 Who this Book is For

This book is intended for IT practitioners who:

- want to learn proven design solutions and practices for building SOA implementations
- want to prepare themselves for common challenges associated with the definition and design of services and service-oriented solutions
- want to learn about SOA and service-orientation by studying detailed aspects of fundamental design
- want to learn about the different types of service-oriented architectures and understand exactly how they are distinct from other architectural models
- want to gain a deep insight into the complexion of modern-day service-oriented solution design

This book can essentially be considered a reference text for use by anyone involved with the construction of service-oriented solutions.

1.3 What this Book Does Not Cover

The following sections highlight specific subject areas not addressed in this book.

Topics Covered by Other Books

This title is dedicated to documenting design patterns only. Because most of the patterns in this catalog were specifically created in support of service-orientation, there are many cross-references to design principles whenever they are related or relevant to a particular pattern. These design principles are covered separately in *SOA Principles of Service Design,* a companion guide for this book.

Furthermore, this book does not contain a tutorial about Web services or service-oriented computing. There are several publications that have already covered these areas in detail. Suggestions are provided in the upcoming *Recommended Reading* section.

Web Service and REST Service Design Patterns

The Web services technology platform has historically influenced the evolution of service-oriented computing, affecting the complexion and feature-set of typical service-oriented architecture implementations. As a result, numerous design patterns in this book make reference to the use of Web services, and the majority of examples provided show services being implemented as Web services.

Furthermore, a series of REST-inspired design patterns were also developed for this pattern catalog but were not considered ready for inclusion in this first edition of the printed *SOA Design Patterns* book. These patterns have been published in the *Candidate* section of the SOAPatterns.org Web site where they will be subjected to on-going reviews, along with other candidate patterns.

It is important to note that the purpose of this book is to provide a catalog of design patterns that help solve problems specific to the realization of SOA and service-orientation. As has been established in previous series titles, Web services and REST services provide implementation *options* for building services as part of service-oriented solutions.

SOA Standardization Efforts

There are several efforts underway by different standards and research organizations to produce abstract definitions, architectural models, and vocabularies for SOA. These projects are in various stages of maturity, and several overlap in scope.

The mandate of this book series is to provide the IT community with current, real-world insight into the most important aspects of service-oriented computing, SOA, and service-orientation. A great deal of research goes into each and every title to follow through on this commitment. This research includes the detailed review of existing and upcoming technologies and platforms, relevant technology products and technology standards, architectural standards and specifications, as well as interviews conducted with key members of leading organizations in the SOA community.

As of the writing of this book there has been no indication that any of the deliverables produced by the aforementioned independent efforts will be adopted as industry-wide SOA standards. In order to maintain an accurate, real-world perspective, these models and vocabularies can therefore not be covered or referenced in this book.

However, given the unpredictable nature of the IT industry, there is always a possibility that one or more of these deliverables will attain industry standard status at some point in time. Should this occur, this book will be supplemented with online content that describes the relationship of the standards to the content of this book and further maps concepts, terms, and models to whatever conventions are established by the standards. This information would be published on SOABooks.com, as described in the *Updates, Errata, and Resources* page. If you'd like to be automatically notified of these types of updates, see the *Notification Service* section for more information.

NOTE

This comment refers to SOA-related specifications only. There are numerous standards initiatives that have produced and continue to produce highly relevant technology specifications, such as those used for XML and Web services. These are referenced, explained, and otherwise documented wherever appropriate in all series titles.

1.4 Recommended Reading

As already mentioned, this book establishes a master pattern catalog for SOA design patterns. Many of these patterns have roots in the following previously published pattern catalogs that are recommended reading, especially if you are new to the world of design patterns:

- *Design Patterns: Elements of Reusable Object-Oriented Software* (E. Gamma, R. Helm, R. Johnson, J. Vlissides, Addison-Wesley 1994)
- *Patterns of Enterprise Application Architecture* (M. Fowler, Addison-Wesley 2003)
- *Enterprise Integration Patterns* (G. Hohpe, B. Woolf, Addison-Wesley 2003)
- *Pattern-Oriented Software Architecture, Volumes 1–5* (F. Buschmann, K. Henney, M. Kircher, R. Meunier, H. Rohnert, D. Schmidt, P. Sommerlad, M. Stal, Wiley 1996–2007)

How the patterns in these and other publications have influenced the SOA design patterns in this book is further discussed in the *Historical Influences* section in Chapter 5.

While this book includes basic patterns that describe foundational parts of SOA and service-orientation in detail, it does not provide a great deal of introductory coverage of SOA or service-oriented computing as a whole. If you are new to SOA and service-orientation, you can consider reading the following titles that are part of this book series:

- *SOA Principles of Service Design*
- *Service-Oriented Architecture: Concepts, Technology, and Design*

Furthermore, in preparation for those patterns that are focused on design solutions that entail the use of Web service technologies, you can use this book as a companion reference guide:

- *Web Service Contract Design and Versioning for SOA*

Note also that the following additional series titles are in development, each of which further explores and builds upon the SOA design patterns documented in this book:

- *SOA with Java*
- *SOA with .NET*
- *ESB Architecture for SOA*
- *SOA Governance*
- *SOA with REST*

You can check on the availability of these titles at SOABooks.com and you can further read up on fundamental topics pertaining to SOA and service-orientation at WhatisSOA.com and SOAPrinciples.com. Finally, you can take advantage of an online master glossary for this book series at SOAGlossary.com.

1.5 How this Book is Organized

This book begins with Chapters 1 and 2 providing introductory content and case study background information respectively. The remainder of the book is grouped into the following parts:

- Part I: Fundamentals
- Part II: Service Inventory Design Patterns
- Part III: Service Design Patterns
- Part IV: Service Composition Design Patterns
- Part V: Supplemental
- Part VI: Appendices

Part I: Fundamentals

Chapters 3, 4, and 5 in this first part set the stage for all of the design patterns that follow in the subsequent parts by covering key terminology and design issues and by providing an exploration of architecture design principles and the service-oriented architecture types that are later referenced by the individual design pattern descriptions. Also provided is an explanation of how pattern profiles in this book are structured, along with additional sections that cover relevant design topics, such as Web services, service design principle references, and pattern types.

Part II: Service Inventory Design Patterns

"Service inventory" is a term used to represent a collection on independently standardized and governed services. Design patterns associated with the design of the service inventory technology architecture are provided in the following chapters:

- *Chapter 6: Foundational Inventory Patterns* – The baseline design characteristics of a service inventory architecture are addressed by a series of closely related design patterns that are presented in a proposed application sequence.
- *Chapter 7: Logical Inventory Layer Patterns* – How services within a service inventory can be grouped into logical layers is covered by a set of design patterns that represent the most common types of service layers.
- *Chapter 8: Inventory Centralization Patterns* – A set of patterns dedicated to centralizing key parts of a service inventory architecture is provided to build upon the preceding fundamental architectural patterns.
- *Chapter 9: Inventory Implementation Patterns* – This more specialized collection of patterns addresses a variety of implementation design issues and options for service inventory architectures.
- *Chapter 10: Inventory Governance Patterns* – Design patterns relating to the post-implementation governance of a service inventory architecture are provided.

Part III: Service Design Patterns

This part is comprised of a set of chapters specific to the design of services and service architecture:

- *Chapter 11: Foundational Service Patterns* – A set of basic design patterns that help establish fundamental service design characteristics via a suggested application sequence. Collectively, these patterns form the most basic application of service-orientation within a service boundary.

- *Chapter 12: Service Implementation Patterns* – A collection of specialized design patterns that provide design solutions for a range of service architecture-specific issues.
- *Chapter 13: Service Security Patterns* – These patterns primarily shape the internal logic of services to equip them with security controls that counter common threats.
- *Chapter 14: Service Contract Design Patterns* – A set of design patterns focused on service contract design concerns both from a contract content and architectural perspective.
- *Chapter 15: Legacy Encapsulation Patterns* – How services can encapsulate and interact with legacy systems and resources is addressed by this set of patterns.
- *Chapter 16: Service Governance Patterns* – For services already deployed and in use, these patterns address common governance issues related to typical post-implementation changes.

Part IV: Service Composition Design Patterns

Service composition design and runtime interaction are addressed by the patterns in the following chapters:

- *Chapter 17: Capability Composition Patterns* – A pair of core patterns that establish the basis of service capability composition as it pertains to composition design and architecture.
- *Chapter 18: Service Messaging Patterns* – This large collection of patterns is focused on inter-service message exchange and processing and provides design solutions for a wide range of messaging concerns.
- *Chapter 19: Composition Implementation Patterns* – Service composition architecture design and runtime composition integrity are addressed by these patterns.
- *Chapter 20: Service Interaction Security Patterns* – A set of patterns focused exclusively on security issues pertaining to runtime service interaction and data exchange.
- *Chapter 21: Transformation Patterns* – Design patterns specific to the runtime transformation of messages via intermediary processing layers.

Part V: Supplemental

- *Chapter 22: Common Compound Design Patterns* – Many of the previously documented design patterns can be combined into compound patterns that solve larger, yet still common design problems. This chapter provides examples of some of the more relevant combinations, including Enterprise Service Bus (704) and Orchestration (701).
- *Chapter 23: Strategic Architecture Considerations* – This chapter essentially provides a strategic context for all of the content covered in previous chapters by revisiting the key goals of service-oriented computing and highlighting how the attainment of each individual goal can impact the different SOA types first established in Chapter 4.
- *Chapter 24: Principles and Patterns at the U.S. Department of Defense* – A brief exploration of how service-orientation design principles and key design patterns are used at the DoD in relation to the Business Operating Environment (BOE).

Part VI: Appendices

- *Appendix A: Case Study Conclusion* – The storylines for the three case studies first introduced in Chapter 2 and then further explored in subsequent chapters are concluded.
- *Appendix B: Candidate Patterns* – The pattern review process is highlighted along with an explanation of how patterns still under review are classified as candidates.
- *Appendix C: Principles of Service-Orientation* – Summarized descriptions of the eight service-orientation design principles are provided for reference purposes.
- *Appendix D: Patterns and Principles Cross-Reference* – This appendix organizes design patterns for quick reference purposes as they pertain to service-orientation design principles.
- *Appendix E: Patterns and Architectural Types Cross-Reference* – Design patterns are cross-referenced with the four service-oriented architecture types established in Chapter 4.

Note that an alphabetical listing of all design patterns together with their page numbers is provided on the inside cover of this book.

1.6 Symbols, Figures, Style Conventions

The books in this series conform to a series of conventions, as explained here.

Symbol Legend

This book contains more than 400 diagrams that are labeled as *figures*. The primary symbols used throughout the figures are individually listed in the symbol legend located on the inside of the front cover.

How Color is Used

Most symbols have distinct colors associated with them so that they are easily recognized within the different figures. One exception to this convention is when portions of a figure need to be highlighted for a particular reason. In this case, symbols may be colored in red. The conflict symbol (which looks like a lightning bolt) is always red because it is used to highlight points of conflict.

Data Flow and Directionality Conventions

Some of the figures in this book deviate from traditional conventions associated with depicting data flow. This is further explained in the *Service Consumer* section in Chapter 3.

Pattern Documentation Conventions

Each pattern in this book is documented in a consistent format according to a set of predefined notation conventions that are explained in the *Pattern Notation* section in Chapter 5. Note that certain general style conventions are changed subsequent to Chapter 5, as explained on the flipside of the Part II divider page (page 110).

1.7 Additional Information

The following sections describe available supplementary information and resources for the books in the *Prentice Hall Service-Oriented Computing Series from Thomas Erl*.

Updates, Errata, and Resources (www.soabooks.com)

Information about other series titles and various supporting resources can be found at SOABooks.com. You are encouraged to visit this site regularly to check for content changes and corrections.

Visio Stencil (www.soabooks.com)

Prentice Hall has produced a Visio stencil containing the color symbols used by the books in this series. This stencil can be downloaded at SOABooks.com.

Community Patterns Site (www.soapatterns.org)

All of the pattern profile summary tables documented in this book are also published online at SOAPatterns.org, as part of an open site for the SOA community dedicated to SOA design patterns. This site allows you to provide feedback regarding any of the design patterns and you can further submit your own pattern candidates. More information about candidate patterns is provided in Appendix B.

Master Glossary (www.soaglossary.com)

This Web site provides a master online glossary for all series titles. The content on this site continues to grow and expand with new glossary definitions as new series titles are developed and released.

Supplementary Posters (www.soaposters.com)

SOAPosters.com provides a set of color posters available for free download as supplements for the books in this series.

The SOA Magazine (www.soamag.com)

The SOA Magazine is a regular publication provided by SOA Systems Inc. and Prentice Hall and is officially associated with the *Prentice Hall Service-Oriented Computing Series from Thomas Erl*. The SOA Magazine is dedicated to publishing specialized SOA articles, case studies, and papers by industry experts and professionals. The common criterion for contributions is that each explores a distinct aspect of service-oriented computing.

Referenced Specifications (www.soaspecs.com)

Various series titles reference or provide tutorials and examples of industry specifications and standards. The SOASpecs.com Web site provides a central portal to the original specification documents created and maintained by the primary standards organizations.

Notification Service

If you'd like to be automatically notified of new book releases in this series, new supplementary content for this title, or key changes to the previously listed Web sites, use the notification form at SOABooks.com.

Contact the Author

To contact me directly, visit my bio site at www.thomaserl.com.

Index of Patterns

Agnostic Capability (Erl), 324
Agnostic Context (Erl), 312
Agnostic Sub-Controller (Erl), 607
Asynchronous Queuing (Little, Rischbeck, Simon), 582
Atomic Service Transaction (Erl), 623
Brokered Authentication (Hogg, Smith, Chong, Hollander, Kozaczynski, Brader, Delgado, Taylor, Wall, Slater, Imran, Cibraro, Cunningham), 661
Canonical Expression (Erl), 275
Canonical Protocol (Erl), 150
Canonical Resources (Erl), 237
Canonical Schema (Erl), 158
Canonical Schema Bus (Utschig, Maier, Trops, Normann, Winterberg, Erl), 709
Canonical Versioning (Erl), 286
Capability Composition (Erl), 521
Capability Recomposition (Erl), 526
Compatible Change (Orchard, Riley), 465
Compensating Service Transaction (Utschig, Maier, Trops, Normann, Winterberg, Loesgen, Little), 631
Composition Autonomy (Erl), 616
Concurrent Contracts (Erl), 421
Contract Centralization (Erl), 409
Contract Denormalization (Erl), 414
Cross-Domain Utility Layer (Erl), 267
Data Confidentiality (Hogg, Smith, Chong, Hollander, Kozaczynski, Brader, Delgado,

Taylor, Wall, Slater, Imran, Cibraro, Cunningham), 641
Data Format Transformation (Little, Rischbeck, Simon), 681
Data Model Transformation (Erl), 671
Data Origin Authentication (Hogg, Smith, Chong, Hollander, Kozaczynski, Brader, Delgado, Taylor, Wall, Slater, Imran, Cibraro, Cunningham), 649
Decomposed Capability (Erl), 504
Decoupled Contract (Erl), 401
Direct Authentication (Hogg, Smith, Chong, Hollander, Kozaczynski, Brader, Delgado, Taylor, Wall, Slater, Imran, Cibraro, Cunningham), 656
Distributed Capability (Erl), 510
Domain Inventory (Erl), 123
Dual Protocols (Erl), 227
Enterprise Inventory (Erl), 116
Enterprise Service Bus (Erl, Little, Rischbeck, Simon), 704
Entity Abstraction (Erl), 175
Event-Driven Messaging (Little, Rischbeck, Simon), 599
Exception Shielding (Hogg, Smith, Chong, Hollander, Kozaczynski, Brader, Delgado, Taylor, Wall, Slater, Imran, Cibraro, Cunningham), 376
Federated Endpoint Layer (Erl), 713
File Gateway (Roy), 457
Functional Decomposition (Erl), 300
Intermediate Routing (Little, Rischbeck, Simon), 549
Inventory Endpoint (Erl), 260
Legacy Wrapper (Erl, Roy), 441
Logic Centralization (Erl), 136
Message Screening (Hogg, Smith, Chong, Hollander, Kozaczynski, Brader, Delgado, Taylor, Wall, Slater, Imran, Cibraro, Cunningham), 381
Messaging Metadata (Erl), 538
Metadata Centralization (Erl), 280
Multi-Channel Endpoint (Roy), 451
Non-Agnostic Context (Erl), 319
Official Endpoint (Erl), 711
Orchestration (Erl, Loesgen), 701

Partial State Deferral (Erl), 356
Partial Validation (Orchard, Riley), 362
Policy Centralization (Erl), 207
Process Abstraction (Erl), 182
Process Centralization (Erl), 193
Protocol Bridging (Little, Rischbeck, Simon), 687
Proxy Capability (Erl), 497
Redundant Implementation (Erl), 345
Reliable Messaging (Little, Rischbeck, Simon), 592
Rules Centralization (Erl), 216
Schema Centralization (Erl), 200
Service Agent (Erl), 543
Service Broker (Little, Rischbeck, Simon), 707
Service Callback (Karmarkar), 566
Service Data Replication (Erl), 350
Service Decomposition (Erl), 489
Service Encapsulation (Erl), 305
Service Façade (Erl), 333
Service Grid (Chappell), 254
Service Instance Routing (Karmarkar), 574
Service Layers (Erl), 143
Service Messaging (Erl), 533
Service Normalization (Erl), 131
Service Perimeter Guard (Hogg, Smith, Chong, Hollander, Kozaczynski, Brader, Delgado, Taylor, Wall, Slater, Imran, Cibraro, Cunningham), 394
Service Refactoring (Erl), 484
State Messaging (Karmarkar), 557
State Repository (Erl), 242
Stateful Services (Erl), 248
Termination Notification (Orchard, Riley), 478
Three-Layer Inventory (Erl), 715
Trusted Subsystem (Hogg, Smith, Chong, Hollander, Kozaczynski, Brader, Delgado, Taylor, Wall, Slater, Imran, Cibraro, Cunningham), 387

UI Mediator (Utschig, Maier, Trops, Normann, Winterberg), 366
Utility Abstraction (Erl), 168
Validation Abstraction (Erl), 429
Version Identification (Orchard, Riley), 472