SOA
Principles of Service Design

Thomas Erl

PRENTICE HALL
UPPER SADDLE RIVER, NJ • BOSTON • INDIANAPOLIS • SAN FRANCISCO
NEW YORK • TORONTO • MONTREAL • LONDON • MUNICH • PARIS • MADRID
CAPETOWN • SYDNEY • TOKYO • SINGAPORE • MEXICO CITY

Contents

Preface xxv

Chapter 1: Introduction 1

1.1 Objectives of this Book 3

1.2 Who this Book Is For 3

1.3 What this Book Does Not Cover 4

Topics Covered by Other Books 4

SOA Standardization Efforts 5

1.4 How this Book Is Organized 6

Part I: Fundamentals 7

Part II: Design Principles 9

Part III: Supplemental 12

Appendices 12

1.5 Symbols, Figures, and Style Conventions 13

Symbol Legend 13

How Color Is Used 13

The Service Symbol 13

1.6 Additional Information 16

Updates, Errata, and Resources (www.soabooks.com) 16

Master Glossary (www.soaglossary.com) 16

Referenced Specifications (www.soaspecs.com) 16

Service-Oriented Computing Poster (www.soaposters.com) 16

The SOA Magazine (www.soamag.com) 17
Notification Service 17
Contact the Author 17

Chapter 2: Case Study 19

2.1 Case Study Background: Cutit Saws Ltd. 20
History 20
Technical Infrastructure and Automation Environment 21
Business Goals and Obstacles 21

PART I: FUNDAMENTALS

Chapter 3: Service-Oriented Computing and SOA. 25

3.1 Design Fundamentals 26
Design Characteristic 27
Design Principle 28
Design Paradigm 29
Design Pattern. 30
Design Pattern Language 31
Design Standard 32
Best Practice 34
A Fundamental Design Framework. 35

3.2 Introduction to Service-Oriented Computing 37
Service-Oriented Architecture. 38
Service-Orientation, Services, and Service-Oriented Solution Logic 39
Service Compositions 39
Service Inventory. 40
Understanding Service-Oriented Computing Elements 40
Service Models 43
SOA and Web Services. 46
Service Inventory Blueprints 51
Service-Oriented Analysis and Service Modeling. 52

Service-Oriented Design . . . 53
Service-Oriented Architecture: Concepts, Technology, and Design . . . 54
3.3 Goals and Benefits of Service-Oriented Computing . . . 55
Increased Intrinsic Interoperability . . . 56
Increased Federation . . . 58
Increased Vendor Diversification Options . . . 59
Increased Business and Technology Domain Alignment . . . 60
Increased ROI . . . 61
Increased Organizational Agility . . . 63
Reduced IT Burden . . . 64
3.4 Case Study Background . . . 66

Chapter 4: Service-Orientation . . . 67

4.1 Introduction to Service-Orientation . . . 68
Services in Business Automation . . . 69
Services Are Collections of Capabilities . . . 69
Service-Orientation as a Design Paradigm . . . 70
Service-Orientation and Interoperability . . . 74
4.2 Problems Solved by Service-Orientation . . . 75
Life Before Service-Orientation . . . 76
The Need for Service-Orientation . . . 81
4.3 Challenges Introduced by Service-Orientation . . . 85
Design Complexity . . . 85
The Need for Design Standards . . . 86
Top-Down Requirements . . . 86
Counter-Agile Service Delivery in Support of Agile Solution Delivery . . . 87
Governance Demands . . . 88
4.4 Additional Considerations . . . 89
It Is Not a Revolutionary Paradigm . . . 89
Enterprise-wide Standardization Is Not Required . . . 89
Reuse Is Not an Absolute Requirement . . . 90

4.5 Effects of Service-Orientation on the Enterprise 91
Service-Orientation and the Concept of "Application" 91
Service-Orientation and the Concept of "Integration" 92
The Service Composition 94
Application, Integration, and Enterprise Architectures 95
4.6 Origins and Influences of Service-Orientation 96
Object-Orientation 97
Web Services 98
Business Process Management (BPM) 98
Enterprise Application Integration (EAI) 98
Aspect-Oriented Programming (AOP) 99
4.7 Case Study Background 100

Chapter 5: Understanding Design Principles 103

5.1 Using Design Principles 104
Incorporate Principles within Service-Oriented Analysis 105
Incorporate Principles within Formal Design Processes 106
Establish Supporting Design Standards 107
Apply Principles to a Feasible Extent 108
5.2 Principle Profiles 109
5.3 Design Pattern References 111
5.4 Principles that Implement vs. Principles that Regulate . . . 111
5.5 Principles and Service Implementation Mediums 114
"Capability" vs. "Operation" vs. "Method" 115
5.6 Principles and Design Granularity 115
Service Granularity 116
Capability Granularity 116
Data Granularity 116
Constraint Granularity 117
Sections on Granularity Levels 118
5.7 Case Study Background 119
The Lab Project Business Process 119

PART II: DESIGN PRINCIPLES

Chapter 6: Service Contracts (Standardization and Design) 125

6.1 Contracts Explained 126
- Technical Contracts in Abstract 126
- Origins of Service Contracts 127

6.2 Profiling this Principle 130

6.3 Types of Service Contract Standardization 132
- Standardization of Functional Service Expression 133
- Standardization of Service Data Representation 134
- Standardization of Service Policies 137

6.4 Contracts and Service Design 140
- Data Representation Standardization and Transformation Avoidance 140
- Standardization and Granularity 142
- Standardized Service Contracts and Service Models 144
- How Standardized Service Contract Design Affects Other Principles 144

6.5 Risks Associated with Service Contract Design 149
- Versioning 149
- Technology Dependencies 150
- Development Tool Deficiencies 151

6.6 More About Service Contracts 152
- Non-Technical Service Contract Documents 152
- "Web Service Contract Design for SOA" 153

6.7 Case Study Example 154
- Planned Services 154
- Design Standards 155
- Standardized WSDL Definition Profiles 155
- Standardized XML Schema Definitions 157
- Standardized Service and Data Representation Layers 157
- Service Descriptions 158
- Conclusion 160

Chapter 7: Service Coupling (Intra-Service and Consumer Dependencies) 163

7.1 Coupling Explained 164
Coupling in Abstract 165
Origins of Software Coupling 165
7.2 Profiling this Principle 167
7.3 Service Contract Coupling Types 169
Logic-to-Contract Coupling (the coupling of service logic to the service contract) 173
Contract-to-Logic Coupling (the coupling of the service contract to its logic) 174
Contract-to-Technology Coupling (the coupling of the service contract to its underlying technology) 176
Contract-to-Implementation Coupling (the coupling of the service contract to its implementation environment) 177
Contract-to-Functional Coupling (the coupling of the service contract to external logic) 180
7.4 Service Consumer Coupling Types 181
Consumer-to-Implementation Coupling 182
Standardized Service Coupling and Contract Centralization . . . 185
Consumer-to-Contract Coupling 185
Measuring Consumer Coupling 191
7.5 Service Loose Coupling and Service Design 193
Coupling and Service-Orientation 193
Service Loose Coupling and Granularity 195
Coupling and Service Models 196
How Service Loose Coupling Affects Other Principles 197
7.6 Risks Associated with Service Loose Coupling 200
Limitations of Logic-to-Contract Coupling 200
Problems when Schema Coupling Is “too loose” 201
7.7 Case Study Example 202
Coupling Levels of Existing Services 202
Introducing the InvLegacyAPI Service 203
Service Design Options 205

Chapter 8: Service Abstraction (Information Hiding and Meta Abstraction Types) 211

8.1 Abstraction Explained 212
Origins of Information Hiding 213
8.2 Profiling this Principle 214
Why Service Abstraction Is Needed 214
8.3 Types of Meta Abstraction 218
Technology Information Abstraction 219
Functional Abstraction 221
Programmatic Logic Abstraction 222
Quality of Service Abstraction 224
Meta Abstraction Types and the Web Service Regions of Influence 225
Meta Abstraction Types in the Real World 227
8.4 Measuring Service Abstraction 231
Contract Content Abstraction Levels 231
Access Control Levels 232
Abstraction Levels and Quality of Service Meta Information . . . 234
8.5 Service Abstraction and Service Design 235
Service Abstraction vs. Service Encapsulation 235
How Encapsulation Can Affect Abstraction 235
Service Abstraction and Non-Technical Contract Documents . . 237
Service Abstraction and Granularity 238
Service Abstraction and Service Models 239
How Service Abstraction Affects Other Principles 239
8.6 Risks Associated with Service Abstraction 242
Multi-Consumer Coupling Requirements 242
Misjudgment by Humans 242
Security and Privacy Concerns 243
8.7 Case Study Example 244
Service Abstraction Levels 244
Operation-Level Abstraction Examples 247

Chapter 9: Service Reusability (Commercial and Agnostic Design) 253

9.1 Reuse Explained 254
Reuse in Abstract 254
Origins of Reuse 257

9.2 Profiling this Principle 259

9.3 Measuring Service Reusability and Applying Commercial Design 262
Commercial Design Considerations 262
Measures of Planned Reuse 265
Measuring Actual Reuse 267
Commercial Design Versus Gold-Plating 267

9.4 Service Reuse in SOA 268
Reuse and the Agnostic Service 268
The Service Inventory Blueprint 269

9.5 Standardized Service Reuse and Logic Centralization . . 270
Understanding Logic Centralization 271
Logic Centralization as an Enterprise Standard 272
Logic Centralization and Contract Centralization 272
Centralization and Web Services 274
Challenges to Achieving Logic Centralization 274

9.6 Service Reusability and Service Design 276
Service Reusability and Service Modeling 276
Service Reusability and Granularity 277
Service Reusability and Service Models 278
How Service Reusability Affects Other Principles 278

9.7 Risks Associated with Service Reusability and Commercial Design 281
Cultural Concerns 281
Governance Concerns 283
Reliability Concerns 286
Security Concerns 286
Commercial Design Requirement Concerns 286
Agile Delivery Concerns 287

9.8 Case Study Example 288
The Inventory Service Profile 288
Assessing Current Capabilities 289
Modeling for a Targeted Measure of Reusability 289
The New EditItemRecord Operation 290
The New ReportStockLevels Operation 290
The New AdjustItemsQuantity Operation 291
Revised Inventory Service Profile 292

Chapter 10: Service Autonomy (Processing Boundaries and Control) 293

10.1 Autonomy Explained 294
Autonomy in Abstract 294
Origins of Autonomy 295
10.2 Profiling this Principle 296
10.3 Types of Service Autonomy 297
Runtime Autonomy (execution) 298
Design-Time Autonomy (governance) 298
10.4 Measuring Service Autonomy 300
Service Contract Autonomy (services with normalized contracts) 301
Shared Autonomy 305
Service Logic Autonomy (partially isolated services) 306
Pure Autonomy (isolated services) 308
Services with Mixed Autonomy 310
10.5 Autonomy and Service Design 311
Service Autonomy and Service Modeling 311
Service Autonomy and Granularity 311
Service Autonomy and Service Models 312
How Service Autonomy Affects Other Principles 314
10.6 Risks Associated with Service Autonomy 317
Misjudging the Service Scope 317
Wrapper Services and Legacy Logic Encapsulation 318
Overestimating Service Demand 318

10.7 Case Study Example 319
Existing Implementation Autonomy of the GetItem Operation . . 319
New Operation-Level Architecture with Increased Autonomy . . 320
Effect on the Run Lab Project Composition 322

Chapter 11: Service Statelessness (State Management Deferral and Stateless Design) 325

11.1 State Management Explained 327
State Management in Abstract 327
Origins of State Management 328
Deferral vs. Delegation 331
11.2 Profiling this Principle 331
11.3 Types of State 335
Active and Passive 335
Stateless and Stateful 336
Session and Context Data 336
11.4 Measuring Service Statelessness 339
Non-Deferred State Management (low-to-no statelessness) . . . 340
Partially Deferred Memory (reduced statefulness) 340
Partial Architectural State Management Deferral (moderate statelessness) 341
Full Architectural State Management Deferral (high statelessness) 342
Internally Deferred State Management (high statelessness) . . . 342
11.5 Statelessness and Service Design 343
Messaging as a State Deferral Option 343
Service Statelessness and Service Instances 344
Service Statelessness and Granularity 346
Service Statelessness and Service Models 346
How Service Statelessness Affects Other Principles 347
11.6 Risks Associated with Service Statelessness 349
Dependency on the Architecture 349
Increased Runtime Performance Demands 350
Underestimating Delivery Effort 350

11.7 Case Study Example 351
Solution Architecture with State Management Deferral 352
Step 1 353
Step 2 354
Step 3 355
Step 4 356
Step 5 357
Step 6 358
Step 7 359

Chapter 12: Service Discoverability (Interpretability and Communication) 361

12.1 Discoverability Explained 362
Discovery and Interpretation, Discoverability and Interpretability in Abstract 364
Origins of Discovery 367
12.2 Profiling this Principle 368
12.3 Types of Discovery and Discoverability Meta Information 371
Design-Time and Runtime Discovery 371
Discoverability Meta Information 373
Functional Meta Data 374
Quality of Service Meta Data 374
12.4 Measuring Service Discoverability 375
Fundamental Levels 375
Custom Rating System 376
12.5 Discoverability and Service Design 376
Service Discoverability and Service Modeling 377
Service Discoverability and Granularity 378
Service Discoverability and Policy Assertions 378
Service Discoverability and Service Models 378
How Service Discoverability Affects Other Principles 378

12.6 Risks Associated with Service Discoverability 381
Post-Implementation Application of Discoverability 381
Application of this Principle by Non-Communicative Resources 381
12.7 Case Study Example 382
Service Profiles (Functional Meta Information) 382
Related Quality of Service Meta Information 386

Chapter 13: Service Composability (Composition Member Design and Complex Compositions) 387

13.1 Composition Explained 388
Composition in Abstract 388
Origins of Composition 390
13.2 Profiling this Principle 392
13.3 Composition Concepts and Terminology 396
Compositions and Composition Instances 397
Composition Members and Controllers 398
Service Compositions and Web Services 401
Service Activities 402
Composition Initiators 403
Point-to-Point Data Exchanges and Compositions 405
Types of Compositions 406
13.4 The Complex Service Composition 407
Stages in the Evolution of a Service Inventory 407
Defining the Complex Service Composition 410
Preparing for the Complex Service Composition 411
13.5 Measuring Service Composability and Composition Effectiveness Potential 412
Evolutionary Cycle States of a Composition 412
Composition Design Assessment 413
Composition Runtime Assessment 415
Composition Governance Assessment 417
Measuring Composability 419

13.6 Composition and Service Design 427
Service Composability and Granularity 427
Service Composability and Service Models 428
Service Composability and Composition Autonomy 430
Service Composability and Orchestration 430
How Service Composability Affects Other Principles 432
13.7 Risks Associated with Service Composition 437
Composition Members as Single Points of Failure 437
Composition Members as Performance Bottlenecks 437
Governance Rigidity of "Over-Reuse" in Compositions 438
13.8 Case Study Example 439

PART III: SUPPLEMENTAL

Chapter 14: Service-Orientation and Object-Orientation: A Comparison of Principles and Concepts 445

14.1 A Tale of Two Design Paradigms 446
14.2 A Comparison of Goals 449
Increased Business Requirements Fulfillment 450
Increased Robustness 451
Increased Extensibility 451
Increased Flexibility 452
Increased Reusability and Productivity 452
14.3 A Comparison of Fundamental Concepts 453
Classes and Objects 453
Methods and Attributes 454
Messages 454
Interfaces 456
14.4 A Comparison of Design Principles 457
Encapsulation 458
Inheritance 459

Generalization and Specialization 461
Abstraction 463
Polymorphism 463
Open-Closed Principle (OCP) 465
Don't Repeat Yourself (DRY) 465
Single Responsibility Principle (SRP) 466
Delegation 468
Association 469
Composition 470
Aggregation 471

14.5 Guidelines for Designing Service-Oriented Classes 472
Implement Class Interfaces 473
Limit Class Access to Interfaces 473
Do Not Define Public Attributes in Interfaces 473
Use Inheritance with Care 473
Avoid Cross-Service "has-a" Relationships 474
Use Abstract Classes for Modeling, Not Design 474
Use Façade Classes 474

Chapter 15: Supporting Practices 477

15.1 Service Profiles 478
Service-Level Profile Structure 478
Capability Profile Structure 480
Additional Considerations 482

15.2 Vocabularies 483
Service-Oriented Computing Terms 484
Service Classification Terms 484
Types and Associated Terms 485
Design Principle Application Levels 487

15.3 Organizational Roles 488
Service Analyst 490
Service Architect 490
Service Custodian 491
Schema Custodian 491
Policy Custodian 492

Service Registry Custodian 492
Technical Communications Specialist 493
Enterprise Architect 493
Enterprise Design Standards Custodian (and Auditor) 494

Chapter 16: Mapping Service-Orientation Principles to Strategic Goals 497

16.1 Principles that Increase Intrinsic Interoperability 498
16.2 Principles that Increase Federation 501
16.3 Principles that Increase Vendor Diversification Options . 501
16.4 Principles that Increase Business and Technology Domain Alignment 502
16.5 Principles that Increase ROI 504
16.6 Principles that Increase Organizational Agility 505
16.7 Principles that Reduce the Overall Burden of IT 507

PART IV: APPENDICES

Appendix A: Case Study Conclusion 513

Appendix B: Process Descriptions 517

B.1 Delivery Processes 518
Bottom-Up vs. Top-Down 518
The Inventory Analysis Cycle 520
Inventory Analysis and Service-Oriented Design 521
Choosing a Delivery Strategy 521
B.2 Service-Oriented Analysis Process 522
Define Analysis Scope 522
Identify Affected Systems 523
Perform Service Modeling 523

B.3 Service Modeling Process 523
B.4 Service-Oriented Design Processes. 525
Design Processes and Service Models 526
Service Design Processes and Service-Orientation 527

Appendix C: Principles and Patterns Cross-Reference 529

Additional Resources 533

About the Author 535

About the Photos 537

Index 539

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 How this Book Is Organized
1.5 Symbols, Figures, and Style Conventions
1.6 Additional Information

Learning from one's mistakes is one of the most essential principles of life. As the old saying goes, "One cannot achieve success without failure." When I hear that saying I sometimes mentally append it with "…unless one happens to be lucky." While there may be some truth to this, the fact is that luck is not something we want to ever have to depend on when building service-oriented architecture (SOA). Optimistic project plans or risk assessments qualified with "…as long as we get lucky" won't have much success instilling confidence (or receiving funding).

A personal mantra of mine that has emerged from involvement in numerous SOA projects preaches that "the key to successfully doing something is in successfully understanding what you're doing." Again, disregarding the luck factor, this philosophy is very relevant to service-oriented computing and forms the basis and purpose of this book.

The content provided in the upcoming chapters is intended to help you become a "true" SOA professional. By that I mean someone who has a clear vision of what it means for a software program to be "service-oriented," who can speak about service-oriented computing from a real-world perspective, and who approaches the design of services with a deep insight into the dynamics behind service-orientation.

Furthermore, such an individual requires the ability to assess options in technology, design, development, delivery, and governance—all important success factors in SOA initiatives. What this translates into for the SOA professional is a need for an increased level of judgment.

Judgment can be seen as a combination of common sense plus a sound knowledge of whatever is being judged. In the world of SOA projects, this points to two specific areas: a need to understand service-oriented computing with absolute clarity and a need to understand your own environments, constraints, and strategic goals just as well. With this range of knowledge, you can leverage what the service-oriented computing platform has to offer in order to fulfill your strategic goals within whatever boundaries you are required to operate.

In theory this makes sense, but there is still something important missing from this formula. Nothing helps raise the level of one's judgment more than actual experience. There's no better way to truly appreciate the strategic potential of service-oriented

computing and the spectrum of challenges that come with its adoption, than to personally go through the motions of a typical enterprise SOA project. This book can't replace real-world experience, but it strives to be the next best thing.

1.1 Objectives of this Book

The focus of this book is first and foremost on the design of services for SOA. There is a constant emphasis on how and where design principles can and should be applied with the ultimate goal of producing high quality services.

Specifically, this book has the following objectives:

- to clearly establish the criteria for solution logic to be classified as "service-oriented"
- to provide complete coverage of the service-orientation design paradigm
- to document specific design characteristics realized by the application of individual design principles
- to describe how the application of each principle affects others
- to explain the link between the design characteristics realized by service-orientation and the strategic goals associated with SOA and service-oriented computing
- to establish the origins of service-orientation and identify how this paradigm differs from other design approaches

Essentially, this guide intends to provide practical, comprehensive, and in-depth coverage of the service-orientation design paradigm, which encompasses the official definition and detailed explanation of eight key principles, each of which is explored in a separate chapter.

1.2 Who this Book Is For

As a guide dedicated to service design, this book will be useful to IT professionals interested in or involved with technology architecture, systems analysis, and solution design.

Specifically, this book will be helpful to developers, analysts, and architects who:

- want to know how to design services for SOA so that they fully support the goals and benefits of service-oriented computing
- want to understand the service-orientation design paradigm

- want to learn about how SOA and service-orientation relate to and can be implemented through Web services
- want in-depth guidance for designing different types of services
- want an understanding of how services need to be designed in support of complex service aggregation and composition
- want to learn about design considerations that apply not just to the entire service, but also to individual service capabilities
- want to better comprehend how services can and should relate to each other
- want deep insight into how service contracts should be shaped in support of service-orientation
- want to know how to determine the appropriate levels of service, capability, data, and constraint granularity
- want an awareness of how WSDL, XML schema, and WS-Policy definitions are best positioned within service designs
- want to understand the origins of service-orientation and how specifically it differs from object-orientation
- will be involved with creating design standards for SOA-based solutions

1.3 What this Book Does Not Cover

SOA and service-oriented computing represent broad subject matters. Many books can be written to explore various aspects of technology, architecture, analysis, and design. This book is focused solely on service engineering and the science of service design.

Topics Covered by Other Books

A primary objective of the *Prentice Hall Service-Oriented Computing Series from Thomas Erl* is to establish a library of complementary books dedicated to service-oriented computing. To accomplish this, an effort has been made to minimize overlap between this title and others in the series.

For example, even though service design touches upon numerous architectural issues, it is important to acknowledge that this is a book about designing services for SOA, not about designing SOA itself. The companion title, *SOA: Design Patterns,* provides a catalog of patterns, many of which deal directly with architectural design.

Furthermore, this book is not a tutorial about Web services or SOA fundamentals. Several books have already covered this ground sufficiently. Although some chapters provide introductory coverage of service-oriented computing, they do not go into detail. A number of sections also assume some knowledge of WSDL, XML schema, and WS-Policy. Basic tutorials for these technologies and structured "how-to" content for SOA is provided in *Service-Oriented Architecture: Concepts, Technology, and Design,* another official companion guide also part of this book series.

Finally, although this book includes a number of case study examples, it does not provide full code samples of implemented services or service contracts. The book *Web Service Contract Design for SOA* is wholly dedicated to the design of Web service contracts and provides both basic and advanced tutorials for WSDL, XML schema, WS-Policy, SOAP, and WS-Addressing. Additionally, several other series titles in development are dedicated to supplying comprehensive coverage of how to build services using different development platforms, such as .NET and Java.

NOTE

There are references to other series titles throughout this book. These references were not added for promotional reasons. In order to establish a well-structured library of complementary books, cross-title references are necessary. They are included for the benefit of the reader to indicate the location of additional relevant resources.

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 some 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 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 an on-going 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 text and further maps the concepts, terms, and models documented in this book to whatever conventions are established by the standards. This information would be published on the corresponding update page, as described in *Updates, Errata, and Resources* section later in this chapter. If you'd like to be automatically notified of these types of updates, see the *Notification Service* section at the end of the chapter for more information.

NOTE

This comment regarding standardization refers to SOA-related specifications only. There are numerous standards initiatives that have and continue to produce highly relevant technology specifications (primarily focused on XML and Web services). These are referenced, explained, and otherwise documented wherever appropriate in all series titles.

1.4 How this Book Is Organized

The organization of content is very straight forward. Chapters 1 and 2 provide background information for the book and its case study, respectively. All subsequent chapters have been grouped into the following primary parts:

- Part I: Fundamentals
- Part II: Design Principles
- Part III: Supplemental

Part I consists of three introductory chapters that set the stage for the detailed exploration of service-orientation design principles provided in Part II. All chapters within these parts communicate primary topics with the assistance of visual style elements and conventions. Diagrams, color, and shading are important style characteristics that have been incorporated to maximize content clarity.

Another means by which additional perspectives are provided is through the use of case study examples. Chapter 2 (which precedes Part I) establishes a case study background from which multiple examples are drawn to supplement the content in subsequent chapters. This supplies a common, real-world context to many of the topics explained in abstract. Up next are brief descriptions of what is covered in subsequent chapters.

Part I: Fundamentals

Although this book is more about applying and realizing service-orientation than it is about understanding SOA basics, we do need to take the time to establish and define key concepts and fundamental terms. These concepts and terms are used throughout the guide, and it is important that their meaning is always clear and consistent. The initial three chapters fulfill this requirement by providing concise introductory coverage.

How these chapters are organized is illustrated in Figure 1.1 and further explained in the upcoming sections.

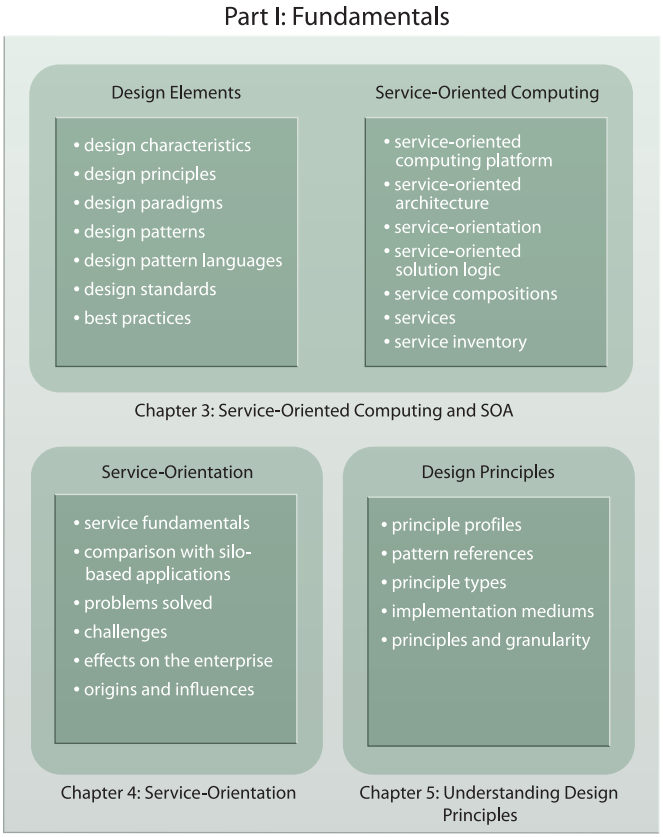


Figure 1.1
The three chapters in Part I deal with the ambiguity surrounding many of the terms and concepts associated with service-oriented computing.

Chapter 3: Service-Oriented Computing and SOA

We begin Part I by establishing the key goals and benefits associated with service-oriented computing. Collectively these goals provide strategic context for all chapters in Part II that document design principles.

This chapter furthermore establishes the service-oriented computing platform by providing definitions for the following terms:

- Service-Oriented Computing
- Service-Oriented Architecture
- Service-Orientation
- Service-Orientation Design Principles
- Service-Oriented Solution Logic
- Services
- Service Compositions
- Service Inventory

In addition to being explained conceptually, the physical relationships of each of these architectural components are also described. The chapter concludes with brief supplemental coverage of additional SOA-related terms, concepts, and processes.

Chapter 4: Service-Orientation

This next chapter zooms in on the design paradigm that underlies service-oriented computing. It begins with an overview of service-orientation by establishing its purpose and goals and then moves on to introduce its eight key design principles. How these principles specifically relate to and support service-oriented architecture is also discussed.

The manner in which the application of service-orientation changes the way solutions are delivered is explored next. Pros and cons of previous approaches are documented and contrasted with the potential for service-orientation to improve upon them. Also explained are the challenges and impositions made by a transition toward this paradigm.

We move on to cover how the adoption of service-orientation transforms not only the technology and the design of an enterprise, but also the mindset and perception of solution logic. Traditional terms, such as "application" and "integration," for example, can be challenged by the fluid nature of service and composition-based automation.

Finally, this introduction ends with a look at some of the key influences of service-orientation. Because this paradigm is very much an evolutionary representation of IT, it is important to acknowledge its roots in past platforms and technology trends.

Chapter 5: Understanding Design Principles

In preparation for Part II, this chapter provides a clear explanation of how subsequent chapters describe service-orientation principles within the context of SOA and service design, and how these principles may relate to design patterns. Different types of principles are categorized, including a study of those that result in implemented design characteristics compared to those that tend to shape and moderate how others are applied. Additionally, four specific forms of contract granularity are established; subsequent chapters then cover how principles affect these granularity types.

Chapter 5 concludes with a case study section that documents a business process for which services will be designed in subsequent chapters.

Part II: Design Principles

Service-orientation is a multi-dimensional subject matter. It is through the application of its design principles that its benefits are realized and that we can build solution logic that can be classified as being truly "service-oriented." This results in an automation environment with unique dynamics and characteristics, all of which need to be understood and planned for.

For example, there are guiding principles that each address a narrow aspect of service design and foster the creation of specific design characteristics. Then there are the issues that arise from combining principles and seeking the right balance for each to be implemented to an appropriate extent.

Part II consists of eight chapters—one for each service-orientation principle, as shown in Figure 1.2. The chapters are structured with a baseline set of sections that are detailed in the *Principle Profiles* section of Chapter 5. Each chapter is further supplemented with a case study example that demonstrates the application of a principle within scenarios drawn from the background established in Chapter 2.

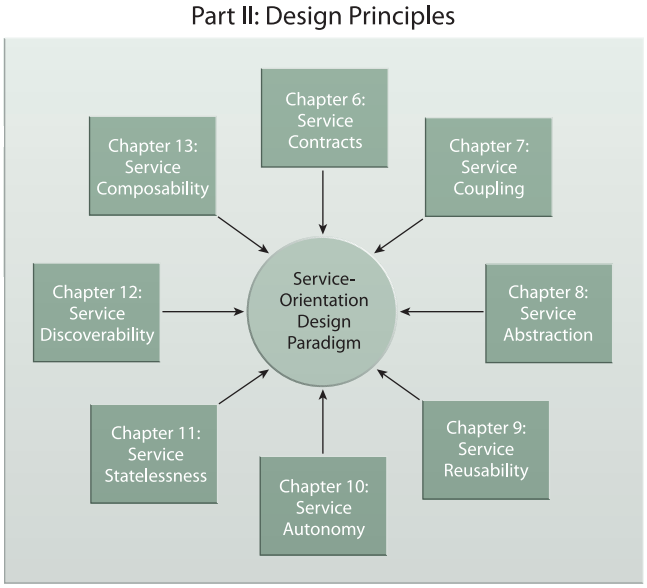


Figure 1.2

A separate chapter is dedicated to exploring each of the eight service-orientation principles. Collectively, these chapters provide a comprehensive documentation of the service-orientation paradigm.

The following sections briefly introduce each chapter:

Chapter 6: Service Contracts (Standardization and Design)

The service contract represents a core part of a service's architecture and is a focal point during the service design process to the extent that a principle is dedicated to its customization. This chapter explains different types of required contract standardization and establishes common levels at which contracts can be harmonized. Issues implicitly introduced by the use of service contracts, such as data models and policies, are discussed, and contracts are further architecturally positioned with an emphasis on Web services.

Chapter 7: Service Coupling (Intra-Service and Consumer Dependencies)

Numerous types of coupling are explored, including the coupling of the service contract to underlying technology and implementation characteristics, as well as the coupling of the service consumers to the contract. This chapter explores levels of attainable coupling and the implications of implementing more or less inter-service dependency. Additionally, the concept of design centralization is introduced as a means of supporting the realization of loose coupling in coordination with other principles.

Chapter 8: Service Abstraction (Information Hiding and Meta Abstraction Types)

The application of this principle determines how much of a service is revealed to the outside world. Achieving a balanced level of abstraction can be one of the most difficult parts of service design. Subsequent to describing the various forms and levels of abstraction, this chapter discusses several associated design risks and the influence abstraction, as a design consideration, has on other principles.

Chapter 9: Service Reusability (Commercial and Agnostic Design)

Increasing the value of solution logic by positioning services as reusable IT assets is a fundamental characteristic and objective of service-orientation. This chapter provides a comprehensive profile of Service Reusability and its implications and extends into an exploration of service reuse levels and the specific influences raised by commercial design considerations. Planned versus actual reuse measuring is discussed, along with the risks and enterprise-wide effects of building and exposing agnostic service logic.

Chapter 10: Service Autonomy (Processing Boundaries and Control)

The ability for a service to have control and governance over its execution environment is key for it to provide reliable, predictable runtime performance, a consideration especially important to the design of service compositions. This chapter explores both runtime and design-time autonomy and provides measurable levels that define an extent of autonomy based on degrees of normalization and functional isolation.

Chapter 11: Service Statelessness (State Management Deferral and Stateless Design)

Service designs capable of deferring state data and state management-related processing enable the implemented service to maximize its availability, an important quality especially in highly concurrent usage environments. Provided in Chapter 11 is a detailed explanation of different types of state information and state management functions followed by levels of attainable service statelessness.

Chapter 12: Service Discoverability (Interpretability and Communication)

The opportunity for services to be utilized to their full potential can only be realized if their existence, purpose, and capabilities are either known or easily located and understood. This chapter focuses on design characteristics associated with the discoverability and interpretability of services as they relate to the overall discovery aspect of service-oriented architecture. A checklist for measuring discoverability is provided, along with sections that document the risks and impacts of discoverability on service models and other principles.

Chapter 13: Service Composability (Composition Member Design and Complex Compositions)

Service composition is a fundamental, yet potentially complex aspect of service-oriented design. This principle deals with it head-on by establishing design requirements to ensure that services can effectively participate in larger composition configurations. A study of how compositions tend to evolve and grow within an enterprise is also provided, along with a series of evaluation criteria to assist in the measuring of a service composition's effectiveness potential.

Part III: Supplemental

Chapter 14: Service-Orientation and Object-Orientation: A Comparison of Principles and Concepts

Object-oriented analysis and design (OOAD) is an established modeling and design paradigm that has influenced numerous aspects of service-orientation. This supplemental comparison is focused on concepts and principles only and is intended for those with an OOAD background.

Chapter 15: Supporting Practices

This next chapter provides a set of supplementary practices and techniques for successfully incorporating and applying service-orientation principles within the common IT enterprise. Specifically, it discusses the use of service profile documents and associated vocabularies, along with common organizational roles.

Chapter 16: Mapping Service-Orientation Principles to Strategic Goals

The book concludes with an exploration of how the eight service-orientation design principles individually relate to and support the strategic goals established in Chapter 3. The content of this final chapter essentially establishes the strategic significance of each design principle.

Appendices

Appendix A: Case Study Conclusion

The case study storyline is concluded here, as the original goals established in Chapter 2 are revisited and assessed against all that transpired in the subsequent case study examples.

Appendix B: Process Descriptions

Service-oriented analysis and design processes are illustrated and briefly described for reference purposes. These processes are explained in detail in the book, *Service-Oriented Architecture: Concepts, Technology, and Design*.

Appendix C: Principles and Patterns Cross-Reference

This last appendix is comprised of a list of design patterns referenced in this book. These patterns are documented separately in the book *SOA: Design Patterns.*

1.5 Symbols, Figures, and Style Conventions

Symbol Legend

This book contains over 240 diagrams, which are referred to as "figures." The primary symbols used throughout all figures are individually described in the symbol legend located on the inside of the front cover.

How Color Is Used

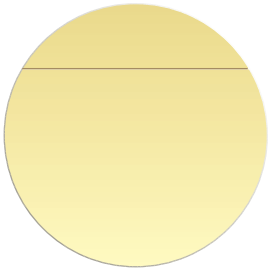
Symbols have distinct colors associated with them so that they are easily recognized within the different figures. The one exception to this convention is when portions of a figure need to be highlighted for a particular reason. In this case, symbols may be colored red. The conflict symbol (which looks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

The Service Symbol

When this book series began, I had already been part of numerous service modeling and design projects during which various tools were (often awkwardly) used to define services and inter-service relationships. I found that it can be beneficial to visually distinguish a technical service contract from other components and systems that also need to be modeled either as parts of the service or as parts of an enterprise environment that need to co-exist with services.

The base symbol I introduced to represent a service throughout the books in this series is a circle divided into two areas (Figure 1.3). This symbol is by no means an industry standard convention. It is only an alternative notation—a means of stating, "This represents something to which we have or intend to apply service-orientation." The remainder of this section provides some background as to how the use of this symbol came about, followed by guidelines.

Figure 1.3
Inspired by the UML class symbol, the service symbol is comprised of two areas wherein the service's name and capabilities are expressed.



Background

In plane geometry, a circle is a highly self-contained form. The use of this shape is appropriate in that it reflects the levels of autonomy, independence, and individuality we seek to establish in every unit of logic we call a service.

This service symbol was recently given an official name: the *chorded circle*, a term coined by Paul Zablosky from the University of British Columbia. This term is also inspired by plane geometry and provides an appropriate metaphor. In the 16th century, mathematician Robert Recorde (also the inventor of the equals "=" sign) wrote, *"If the line goe crosse the circle, and passe beside the centre, then is it called a corde…"* Circles with chords look very much like the symbols in Figure 1.4.

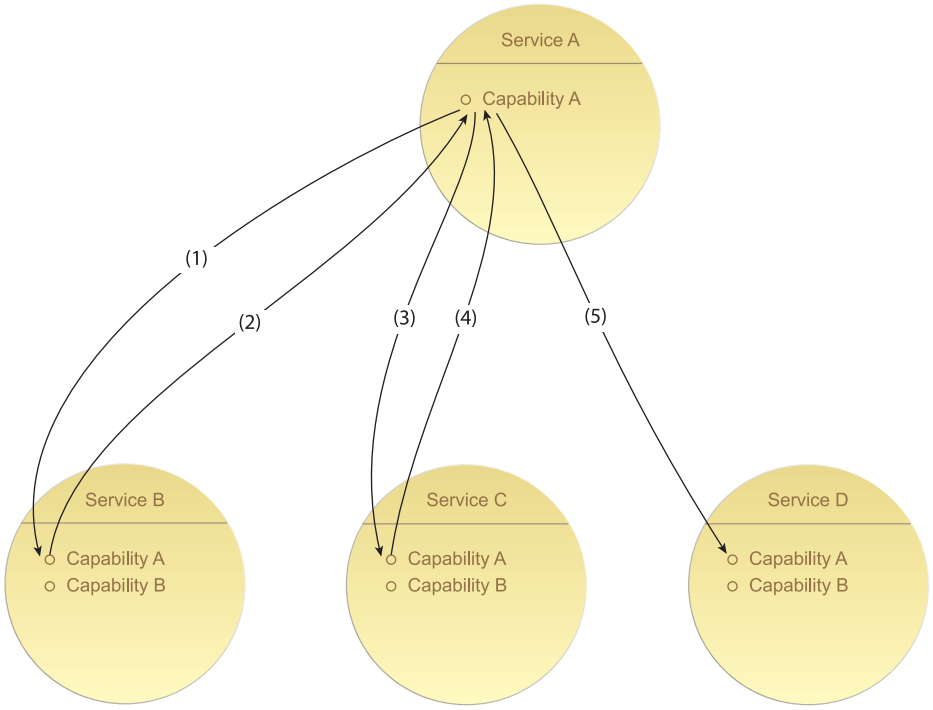


Figure 1.4
A service composition expressed using the chorded circle notation.

When using the chorded circle (or any supplemtary notation you may decide on), the following guidelines are recommended:

The Chorded Circle is an Abstract and Implementation-Neutral Expression of a Service

This symbol does not imply that a service exists as a component or Web service. The symbol simply abstracts the official public technical contract details to establish an official service endpoint definition and to also represent interface details made available to the outside world.

Throughout this book, chorded circles express services with no hint of how the services are actually implemented. Different symbols are used to illustrate physical implementation details of services as components and Web services. (These symbols are explained in the symbol legend mentioned previously.)

The Chorded Circle Is Complementary to UML

As explained in Chapter 14, this symbol can be used on its own to represent abstract technical service contracts, it can be used in conjunction with traditional UML notation, or it does not need to be used at all. Portions of UML can be adapted and used instead to express technical service contract details.

The Chorded Circle Represents a Member of a Service Inventory

What is most important about what this symbol visually communicates is that it represents a unit of logic designed as a service. In other words, it is not used to represent just a Web service or a component, but an actual service shaped by service-orientation and part of a larger collective known as a service inventory (as explained in Chapter 3).

The Basic Chorded Circle Is Most Useful for Modeling Purposes

The base version of this symbol does not provide a great deal of detail about the service contract. It will therefore get you only so far within a service delivery lifecycle. Its primary usage is during the service-oriented analysis process during which service modeling is carried out and service candidates are collaboratively defined and repeatedly refined by business and technology experts as part of a service inventory blueprint.

The Chorded Circle Notation Is Extensible

While the base version of this symbol provides only a simple, abstract expression of a service, extended versions can be created with more detail. Additional labels and

qualifiers are available to express further service characteristics, such as message exchange patterns, policy assertions, service models, implementation and encapsulation characteristics, and lifecycle status. However, to keep things simple, these extensions are not used in this book.

1.6 Additional Information

The following sections provide supplementary information and resources for 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 `www.soabooks.com`. I would encourage you to visit the update page for this book regularly to check for content changes and corrections. I periodically review and revise book content to reflect industry developments.

Master Glossary (www.soaglossary.com)

To avoid content overlap and to ensure constant content currency, the books in this series do not contain glossaries. Instead, a dedicated Web site at `www.soaglossary.com` provides a master glossary for all series titles. This site continues to grow and expand with new glossary definitions as new series titles are developed and released.

Referenced Specifications (www.soaspecs.com)

Various series titles reference or provide tutorials and examples of open XML and Web services specifications and standards. The `www.soaspecs.com` Web site provides a central portal to the original specification documents created and maintained by the primary standards organizations.

Service-Oriented Computing Poster (www.soaposters.com)

The inside of the front cover contains a collection of diagrams for quick reference purposes. A separate color reference poster including these and additional illustrations and content is also available. Visit `www.soaposters.com` for more information.

The SOA Magazine (www.soamag.com)

The SOA Magazine is a regular publication provided by SOA Systems Inc. and Prentice Hall/PearsonPTR 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 criteria for contributions is that each explores a distinct aspect of service-oriented computing.

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, send a blank e-mail to `notify@soabooks.com`.

Contact the Author

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

About the Author

Thomas Erl is the world's top-selling SOA author, the Series Editor of the *Prentice Hall Service-Oriented Computing Series from Thomas Erl*, and Editor of *The SOA Magazine*.

With over 65,000 copies in print, his first two books, *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services* and *Service-Oriented Architecture: Concepts, Technology, and Design* have become international bestsellers and have been translated into several languages. Books by Thomas Erl have been formally reviewed and endorsed by senior members of major software organizations, including IBM, Sun, Microsoft, Oracle, BEA, HP, SAP, Google, and Intel.

Thomas is also the founder of SOA Systems Inc. (`www.soasystems.com`), a company specializing in SOA training and strategic consulting services with a vendor-agnostic focus. Through his work with standards organizations and independent research efforts, Thomas has made significant contributions to the SOA industry, most notably in the areas of service-orientation and SOA methodology.

Thomas is a speaker and instructor for private and public events, and has delivered many workshops and keynote speeches. For a current list of his workshops, seminars, and courses, see `www.soatraining.com`.

Papers and articles written by Thomas have been published in numerous industry trade magazines and Web sites, and he has delivered Webcasts and interviews for many publications, including the *Wall Street Journal*.

For more information, visit `www.thomaserl.com`.

THE PRENTICE HALL SERVICE-ORIENTED COMPUTING SERIES FROM THOMAS ERL

Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services

ISBN 0131428985

This top-selling field guide offers expert advice for incorporating XML and Web services technologies within service-oriented integration architectures.

Service-Oriented Architecture: Concepts, Technology, and Design

ISBN 0131858580

Widely regarded as the definitive "how-to" guide for SOA, this best-selling book presents a comprehensive end-to-end tutorial that provides step-by-step instructions for modeling and designing service-oriented solutions from the ground up.

SOA: Principles of Service Design

ISBN 0132344823

Published with over 240 color illustrations, this hands-on guide contains practical, comprehensive, and in-depth coverage of service engineering techniques and the service-orientation design paradigm. Proven design principles are documented to help maximize the strategic benefit potential of SOA.

SOA: Design Patterns

ISBN 0136135161

Software design patterns have emerged as a powerful means of avoiding and overcoming common design problems and challenges. This new book presents a formal catalog of design patterns specifically for SOA and service-orientation. All patterns are documented using full-color illustrations and further supplemented with case study examples.

Several additional series titles are currently in development and will be released soon. For more information about any of the books in this series, visit www.soabooks.com.

PRENTICE HALL