

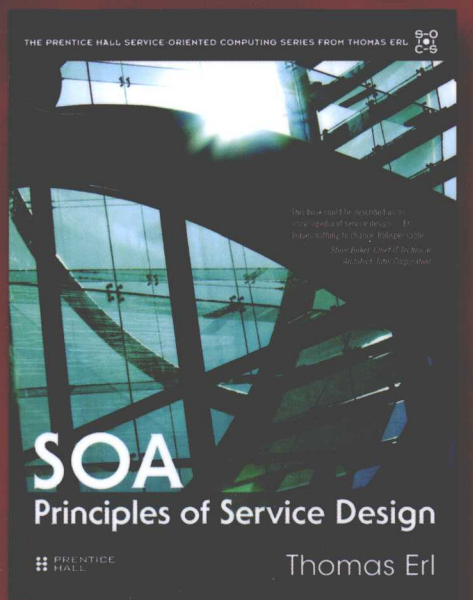
服 务 计 算 技 术 丛 书

PEARSON

SOA服务设计原则

(英文版)

[美] Thomas Erl 著



SOA: Principles of Service Design



科学出版社

PEARSON

服务计算技术丛书

SOA 服务设计原则

(英文版)

SOA: Principles of Service Design

〔美〕 Thomas Erl 著

科学出版社

北 京

图字: 01-2012-0460

内 容 简 介

成功使用面向服务架构(SOA)的关键在于理解其最基本的组成模块——服务的含义和重要性。本书首先简要介绍了 SOA 与服务计算的概念和特点,然后着重阐述了 8 个核心设计原则:标准化服务合约、服务松散耦合、服务抽象、服务可复用性、服务自治、服务无状态性、服务可发现性和服务可组合性,每个设计原则都附有详细的设计范例。全书结构清晰、深入浅出,而且附有与《SOA 设计模式》中关键设计模式之间的交叉参考。通过学习本书,读者能够学会如何设计现实中的 SOA。

本书可供 SOA 领域的软件架构师、高级软件工程师、分析师、应用科研人员等参考学习。

Original edition, entitled SOA: PRINCIPLES OF SERVICE DESIGN, 1E, 9780132344821 by ERL, THOMAS, published by Pearson Education, Inc, publishing as Prentice Hall, Copyright © 2008 SOA Systems, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

China edition published by PEARSON EDUCATION ASIA LTD., and CHINA SCIENCE PUBLISHING & MEDIA LTD. (SCIENCE PRESS) Copyright © 2012.

Authorized for sale and distribution in the People's Republic of China exclusively (except Taiwan, Hong Kong SAR and Macau SAR). 本版本仅可在中国地区(不包括香港、澳门和台湾)销售发行。

本书封面贴有 Pearson Education (培生教育出版集团)激光防伪标签。无标签者不得销售。

图书在版编目(CIP)数据

SOA 服务设计原则 = SOA: Principles of Service Design: 英文 / (美) 艾尔 (Erl, T.) 著.
—北京: 科学出版社, 2012

(服务计算技术丛书)

ISBN 978-7-03-033602-6

I. ①S… II. ①艾… III. ①互联网络—网络服务器—研究—英文 IV. ①TP368.5

中国版本图书馆 CIP 数据核字(2012)第 027553 号

责任编辑: 朱雪玲 / 责任印制: 赵 博 / 封面设计: 迷底书装

科学出版社出版

北京东黄城根北街16号

邮政编码: 100717

<http://www.sciencep.com>

双青印刷厂印刷

科学出版社发行 各地新华书店经销

*

2012 年 3 月第 一 版 开本: B5(720×1000)

2012 年 3 月第一次印刷 印张: 37 1/2

字数: 598 000

定价: 95.00 元

(如有印装质量问题, 我社负责调换)

Preface

Over the past few years I've been exposed to many different IT environments as part of a wide range of SOA initiatives for clients in both private and public sectors. While doing some work on a project for a client in the defense industry, I had an opportunity to learn more about not just their technical landscape, but also the various policies and procedures that are specific to the defense culture. During this time I came across the DoD Standardization Program, an initiative comprised of documents and specifications that establish guiding principles and standards for various aspects of the military, including the design of weapons and military equipment, as well as the definition of methods and processes used by military personnel.

While reading about this program, I learned that several other standardization programs have been in existence for some time, facilitating standardization within public sector organizations (such as the Coast Guard and NASA), as well as numerous private sector industries. The goals of these programs tend to revolve around the establishment of industry standards to enhance interoperability with the ultimate objective of reducing operational overhead, reducing risk, and increasing the organization's overall effectiveness.

In the case of the aforementioned public sector-related standards, interoperability may refer to the exchange of equipment or weapons or the exchange and collaboration of personnel from different locations.

For example, an ammunition clip manufactured in Iowa, stored in Virginia, and delivered to and used by someone at a training base in Texas will work perfectly with a gun manufactured in Kansas because both of these products were built according to the same set of specifications. Similarly, in response to a natural disaster a rescue team may

need to be quickly assembled from individuals based out of different cities and who have never previously worked together. This team can still function effectively because all team members were trained as per the same procedures and processes, using the same vocabulary and conventions.

These standardization programs have much in common with the rationale and objectives behind SOA and service-orientation. The fundamental goal is to produce something with repeatable value, long-term benefit, and inherent flexibility, all for the strategic good of the organization. The greatest obstacle to achieving this goal in the world of SOA has been a lack of understanding as to what service-orientation, as an industry paradigm, really is. It is my hope that this book will help rectify this situation by providing some clarity for what it means for something to be “service-oriented.”

Contents

Preface	xvii
--------------------------	-------------

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-Oriented Architecture, 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