

<<SOA服务设计原则>>

图书基本信息

书名：<<SOA服务设计原则>>

13位ISBN编号：9787030336026

10位ISBN编号：703033602X

出版时间：2012-3

出版时间：科学出版社

作者：Thomas Erl

页数：577

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

<<SOA服务设计原则>>

内容概要

本书主要介绍了SOA基础和SOA设计原则，包括服务协议、服务耦合、服务抽象、服务可重用、服务自治、服务状态管理、服务发现、服务组合的设计原则和应用案例，最后对SOA和面向对象的设计方法进行了对比，在附录中给出SOA的服务交付、分析、服务建模、服务设计等参考流程。本书对业务工程进行了彻底的研究，引领读者学习了综合的、深入的、可视化的面向服务设计范例，精确地揭示了现实中的SOA服务应该如何设计。

<<SOA服务设计原则>>

作者简介

作者：(美国)Thomas Erl

<<SOA服务设计原则>>

书籍目录

Preface
 Chapter 1: Introduction
 1.1 Objectives of this Book
 1.2 Who this Book Is For
 1.3 What this Book Does Not Cover
 Topics Covered by Other Books
 SOA Standardization Efforts
 1.4 How this Book Is Organized
 Part I: Fundamentals
 Part II: Design Principles
 Part III: Supplemental
 Appendices
 1.5 Symbols, Figures, and Style Conventions
 Symbol Legend
 How Color Is Used
 The Service Symbol
 1.6 Additional Information
 Updates, Errata, and Resources (www.soabooks.com)
 Master Glossary (www.soaglossary.com)
 Referenced Specifications (www.soaspecs.com)
 Service-Oriented Computing Poster (www.soaposters.com)
 The SOA Magazine (www.soamag.com)
 Notification Service
 Contact the Author
 Chapter 2: Case Study
 2.1 Case Study Background: Cutit Saws Ltd
 History
 Technical Infrastructure and Automation Environment
 Business Goals and Obstacles
 PART I: FUNDAMENTALS
 Chapter 3: Service-Oriented Computing and SOA
 3.1 Design Fundamentals
 Design Characteristic
 Design Principle
 Design Paradigm
 Design Pattern
 Design Pattern Language
 Design Standard
 Best Practice
 A Fundamental Design Framework
 3.2 Introduction to Service-Oriented Computing
 Service-Oriented Architecture
 Service-Oriented Architecture, Services, and Service-Oriented Solution
 Logic
 Service Compositions
 Service Inventory
 Understanding Service-Oriented Computing Elements
 Service Models
 SOA and Web Services
 Service Inventory Blueprints
 Service-Oriented Analysis and Service Modeling
 Service-Oriented Design
 Service-Oriented Architecture: Concepts, Technology, and Design
 3.3 Goals and Benefits of Service-Oriented Computing
 Increased Intrinsic Interoperability
 Increased Federation
 Increased Vendor Diversification Options
 Increased Business and Technology Domain Alignment
 Increased ROI
 Increased Organizational Agility
 Reduced IT Burden
 3.4 Case Study Background
 Chapter 4: Service-Oriented Computing
 4.1 Introduction to Service-Oriented Computing
 Services in Business Automation
 Services Are Collections of Capabilities
 Service-Oriented Computing as a Design Paradigm
 Service-Oriented Computing and Interoperability
 4.2 Problems Solved by Service-Oriented Computing
 Life Before Service-Oriented Computing
 The Need for Service-Oriented Computing
 4.3 Challenges Introduced by Service-Oriented Computing
 Design Complexity
 The Need for Design Standards
 Top-Down Requirements
 Counter-Agile Service Delivery in Support of Agile Solution Delivery
 Governance Demands
 4.4 Additional Considerations
 It Is Not a Revolutionary Paradigm
 Enterprise-wide Standardization Is Not Required
 Reuse Is Not an Absolute Requirement
 4.5 Effects of Service-Oriented Computing on the Enterprise
 Service-Oriented Computing and the Concept of "Application"
 Service-Oriented Computing and the Concept of "Integration"
 The Service Composition
 Application, Integration, and Enterprise Architectures
 4.6 Origins and Influences of Service-Oriented Computing
 Object-Oriented Computing
 Web Services
 Business Process Management (BPM)
 Enterprise Application Integration (EAI)
 Aspect-Oriented Programming (AOP)
 4.7 Case Study Background
 Chapter 5: Understanding Design Principles
 5.1 Using Design Principles
 Incorporate Principles within Service-Oriented Analysis
 Incorporate Principles within Formal Design Processes
 Establish Supporting Design Standards
 Apply Principles to a Feasible Extent
 5.2 Principle Profiles
 5.3 Design Pattern References
 5.4 Principles that Implement vs. Principles that Regulate
 5.5 Principles and Service Implementation Mediums
 "Capability" vs. "Operation" vs. "Method"
 5.6 Principles and Design Granularity
 Service Granularity
 Capability Granularity
 Data Granularity
 Constraint Granularity
 Sections on Granularity Levels
 5.7 Case Study Background
 The Lab Project Business Process
 PART II: DESIGN PRINCIPLES
 Chapter 6: Service Contracts (Standardization and Design)
 6.1 Contracts Explained
 Technical Contracts in Abstract
 Origins of Service Contracts
 6.2 Profiling this Principle
 6.3 Types of Service Contract
 Standardization
 Standardization of Functional Service Expression
 Standardization of Service Data Representation
 Standardization of Service Policies
 6.4 Contracts and Service Design
 Data Representation
 Standardization and Transformation Avoidance
 Standardization and Granularity
 Standardized Service Contracts and Service Models
 How Standardized Service Contract Design Affects Other Principles
 6.5 Risks Associated with Service Contract Design
 Versioning
 Technology Dependencies
 Development Tool Deficiencies
 6.6 More About Service Contracts
 Non-Technical Service Contract Documents
 "Web Service Contract Design for SOA"
 6.7 Case Study Example
 Planned Services
 Design Standards
 Standardized WSDL Definition Profiles
 Standardized XML Schema Definitions
 Standardized Service and Data Representation Layers
 Service Descriptions
 Conclusion
 Chapter

<<SOA服务设计原则>>

7:Service Coupling(Intra-Service and Consumer Dependencies)7.1 Coupling ExplainedCoupling in AbstractOrigins of Software Coupling7.2 Profiling this Principle7.3 Service Contract Coupling TypesLogic-to-Contract Coupling(the coupling of service logic to the service contract)Contract-to-Logic Coupling(the coupling of the service contract to its logic)Contract-to-Technology Coupling(the coupling of the service contract to its underlying technology)Contract-to-Implementation Coupling(the coupling of the service contract to its implementation environment)Contract-to-Functional Coupling(the coupling of the service contract to external logic)7.4 Service Consumer Coupling TypesConsumer-to-Implementation CouplingStandardized Service Coupling and Contract CentralizationConsumer-to-Contract CouplingMeasuring Consumer Coupling7.5 Service Loose Coupling and Service DesignCoupling and Service-Oriented Service Loose Coupling and GranularityCoupling and Service ModelsHow Service Loose Coupling Affects Other Principles7.6 Risks Associated with Service Loose CouplingLimitations of Logic-to-Contract CouplingProblems when Schema Coupling Is "too loose"7.7 Case Study ExampleCoupling Levels of Existing ServicesIntroducing the InvLegacyAPI ServiceService Design OptionsChapter 8:Service Abstraction(Information Hiding and Meta Abstraction Types)8.1 Abstraction ExplainedOrigins of Information Hiding8.2 Profiling this PrincipleWhy Service Abstraction Is Needed8.3 Types of Meta AbstractionTechnology Information AbstractionFunctional AbstractionProgrammatic Logic AbstractionQuality of Service AbstractionMeta Abstraction Types and the Web Service Regions of InfluenceMeta Abstraction Types in the Real World8.4 Measuring Service AbstractionContract Content Abstraction LevelsAccess Control LevelsAbstraction Levels and Quality of Service Meta Information8.5 Service Abstraction and Service DesignService Abstraction vs.Service EncapsulationHow Encapsulation Can Affect AbstractionService Abstraction and Non-Technical Contract DocumentsService Abstraction and GranularityService Abstraction and Service ModelsHow Service Abstraction Affects Other Principles8.6 Risks Associated with Service AbstractionMulti-Consumer Coupling RequirementsMisjudgment by HumansSecurity and Privacy Concerns8.7 Case Study ExampleService Abstraction LevelsOperation-Level Abstraction ExamplesChapter 9:Service Reusability(Commercial and Agnostic Design)9.1 Reuse ExplainedReuse in AbstractOrigins of Reuse9.2 Profiling this Principle9.3 Measuring Service Reusability and Applying Commercial DesignCommercial Design ConsiderationsMeasures of Planned ReuseMeasuring Actual ReuseCommercial Design Versus Gold-Plating9.4 Service Reuse in SOAReuse and the Agnostic ServiceThe Service Inventory Blueprint9.5 Standardized Service Reuse and Logic CentralizationUnderstanding Logic CentralizationLogic Centralization as an Enterprise StandardLogic Centralization and Contract CentralizationCentralization and Web ServicesChallenges to Achieving Logic Centralization9.6 Service Reusability and Service DesignService Reusability and Service ModelingService Reusability and GranularityService Reusability and Service ModelsHow Service Reusability Affects Other Principles9.7 Risks Associated with Service Reusability and Commercial DesignCultural ConcernsGovernance ConcernsReliability ConcernsSecurity ConcernsCommercial Design Requirement ConcernsAgile Delivery Concerns9.8 Case Study ExampleThe Inventory Service ProfileAssessing Current CapabilitiesModeling for a Targeted Measure of ReusabilityThe New EditItemRecord OperationThe New ReportStockLevels OperationThe New AdjustItemsQuantity OperationRevised Inventory Service ProfileChapter 10:Service Autonomy(Processing Boundaries and Control)10.1 Autonomy ExplainedAutonomy in AbstractOrigins of Autonomy10.2 Profiling this Principle10.3 Types of Service AutonomyRuntime Autonomy(execution)Design-Time Autonomy(governance)10.4 Measuring Service AutonomyService Contract Autonomy(services with normalized contracts)Shared AutonomyService Logic Autonomy(partially isolated services)Pure Autonomy(isolated services)Services with Mixed Autonomy10.5 Autonomy and Service DesignService Autonomy and Service Modeling

章节摘录

版权页: 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

<<SOA服务设计原则>>

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

Aprimary 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.

1.3 What this Book Does Not Cover

5Furthermore, 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 WSPolicy. 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.

<<SOA服务设计原则>>

编辑推荐

《SOA服务设计原则(英文版)》对业务工程进行了彻底的研究,引领读者学习了综合的、深入的、可视化的面向服务设计范例,精确地揭示了现实中的SOA服务应该如何设计。
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。

<<SOA服务设计原则>>

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>