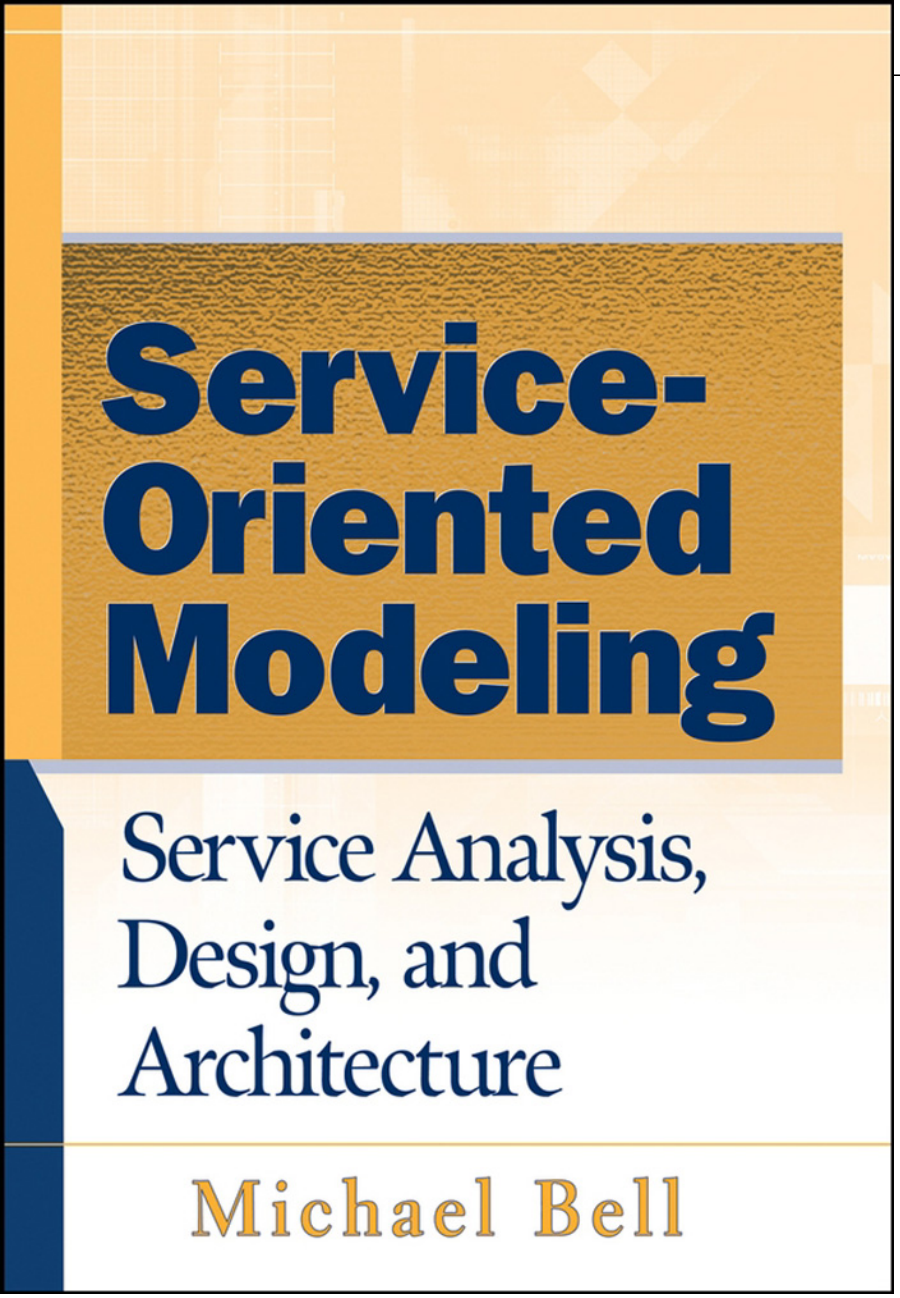

Service-Oriented Modeling
Service Analysis, Design, and Architecture
Michael Bell

SERVICE-ORIENTED MODELING

SERVICE ANALYSIS, DESIGN, AND ARCHITECTURE

MICHAEL BELL

JOHN WILEY & SONS, INC.

SERVICE-ORIENTED MODELING

SERVICE ANALYSIS, DESIGN, AND ARCHITECTURE

MICHAEL BELL

JOHN WILEY & SONS, INC.

This book is printed on acid-free paper.

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.

Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data:

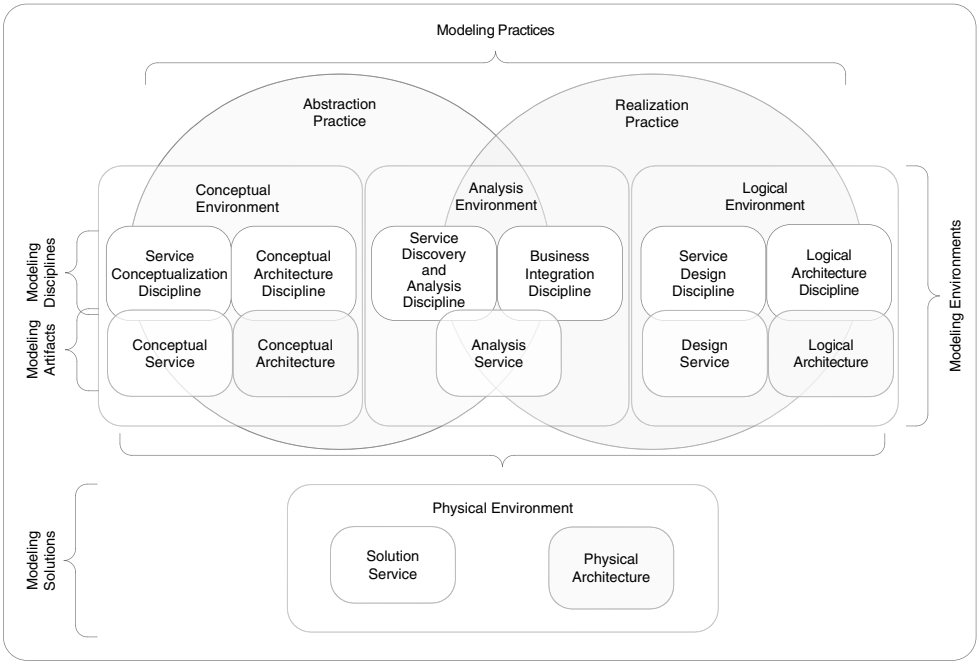
Bell, Michael, 1951—
Service-oriented modeling : service analysis, design, and architecture / Michael Bell.
p. cm.
Includes index.
ISBN 978-0-470-14111-3 (cloth)
1. Business enterprises—Computer networks—Management. 2. System design. 3. Computer software—Development. 4. Computer simulation. 5. Computer network architectures. I. Title.
HD30.37.B45 2008
004.068—dc22

2007033463

Printed in the United States of America.

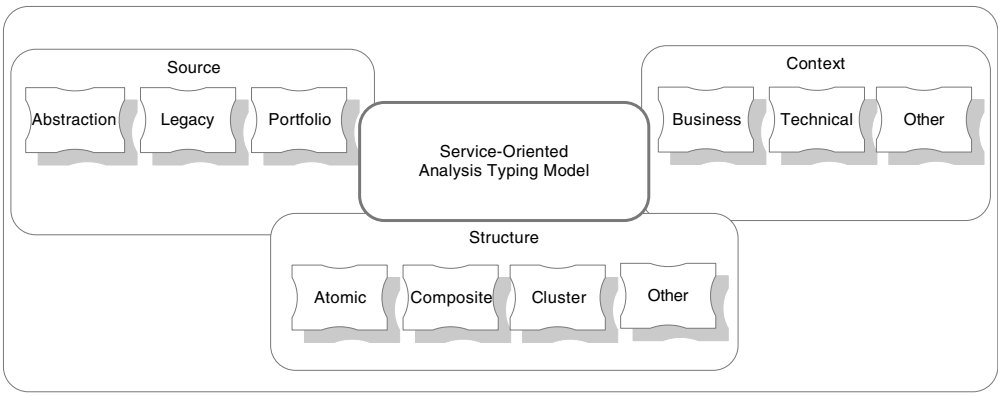
10 9 8 7 6 5 4 3 2 1

For Yvonne, whose love, patience, and support carried me through this project.



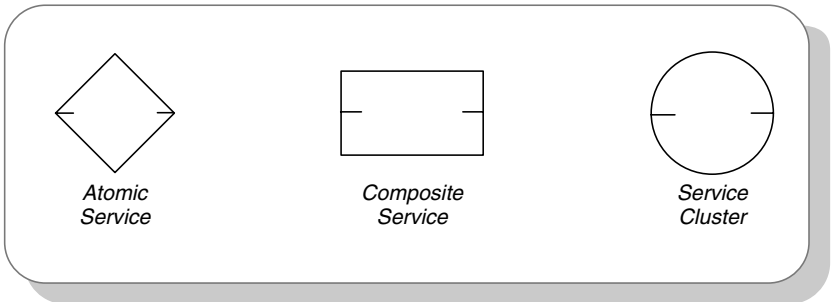
Service-Oriented Modeling Framework

Service Typing Model

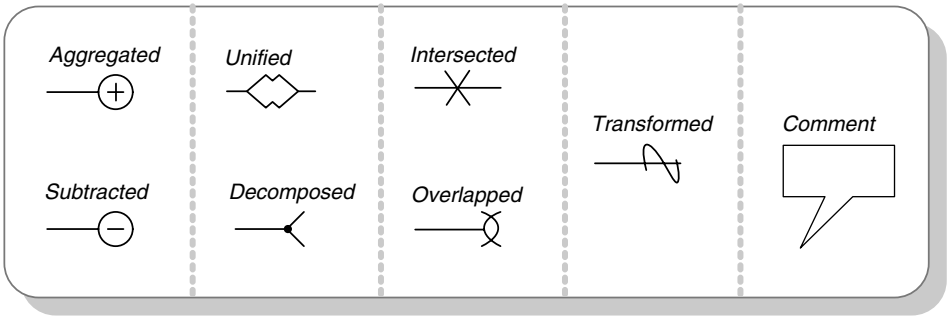


Service-Oriented Analysis Typing Model

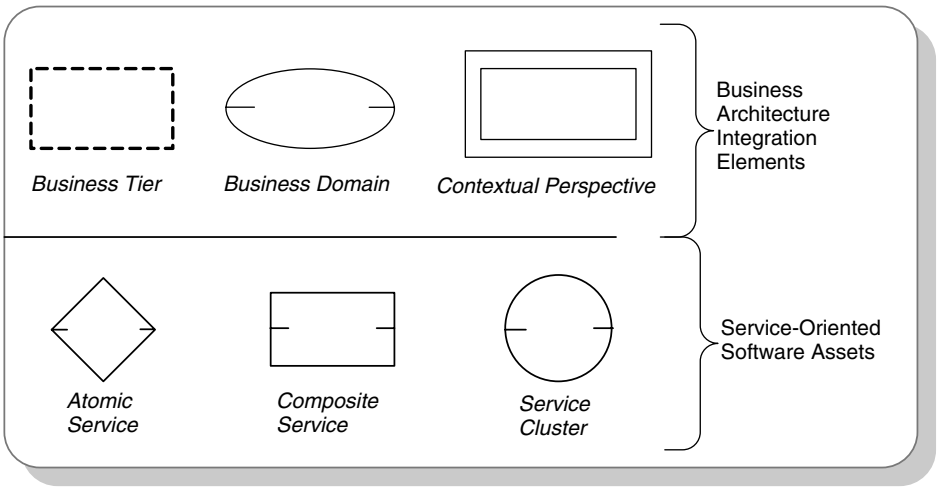
Service-Oriented Analysis Notation



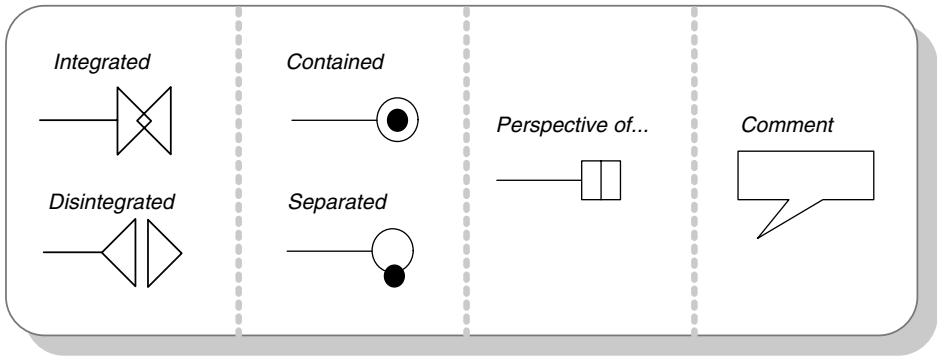
Service-Oriented Analysis Asset Notation



Service-Oriented Analysis Operation Notation

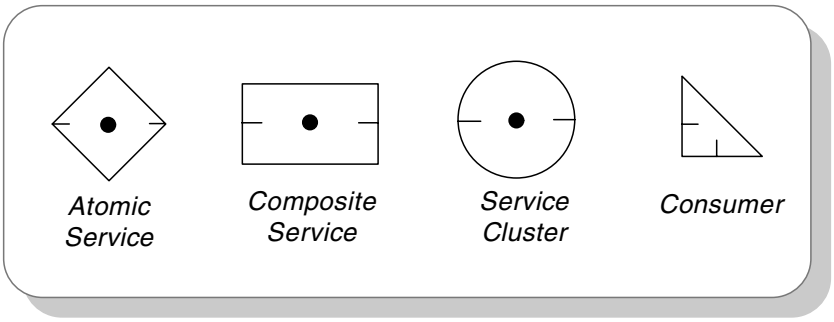


Service-Oriented Business Integration Asset Notation

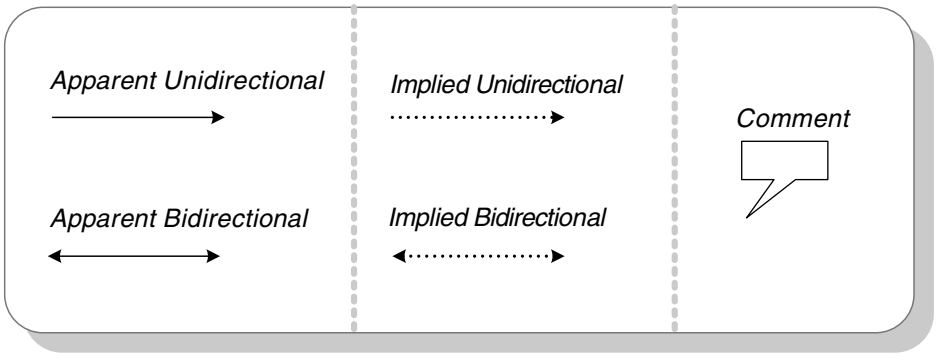


Service-Oriented Business Integration Operations Notation

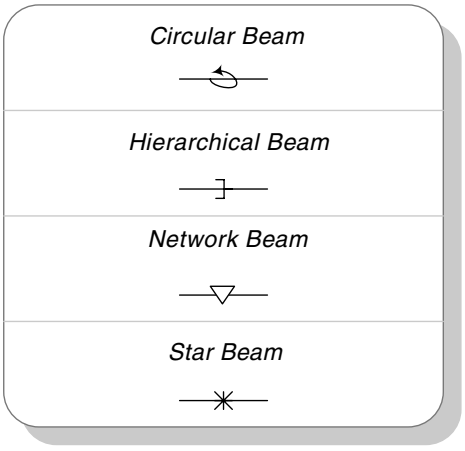
Service-Oriented Design Notation



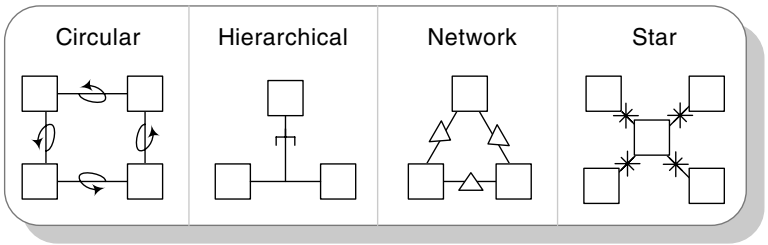
Service-Oriented Design Asset Notation



Service-Oriented Logical Design Relationship Connectors

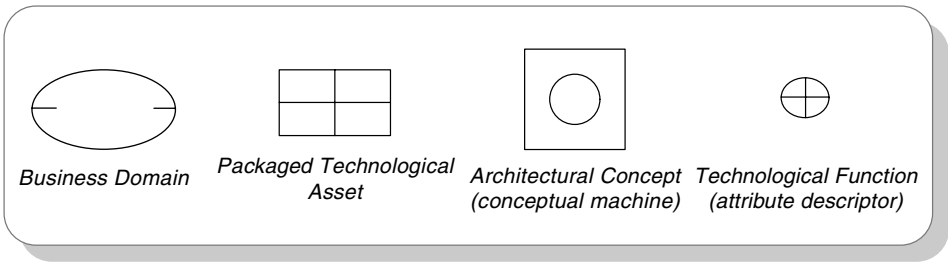


Service-Oriented Design Composition Style Beams

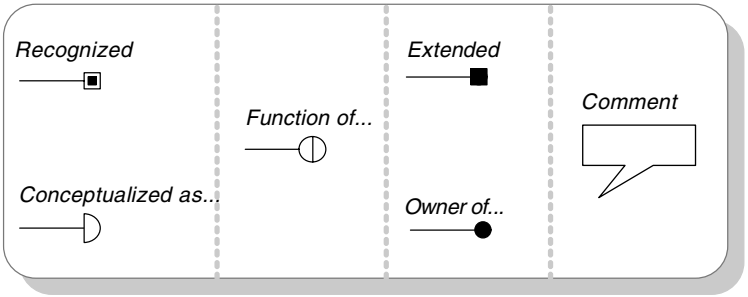


Logical Design Composition Styles

Service-Oriented Conceptual Architecture Notation

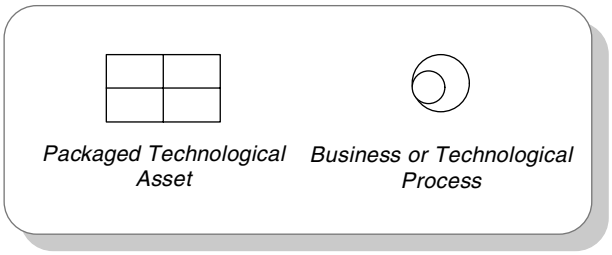


Conceptual Architecture Solution Elements

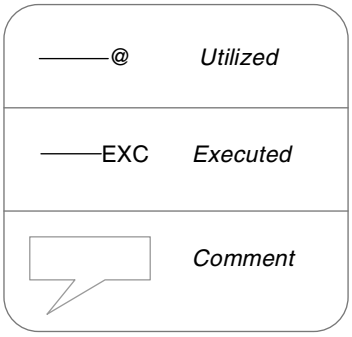


Conceptual Architecture Operations Notation

Service-Oriented Logical Architecture Notation



Logical Architecture Assets Notation



Logical Architecture Operations Notation

CONTENTS

PREFACE

Not long after the beginning of the current decade, the new service-oriented architecture (SOA) paradigm picked up steam and was established as a leading business and technology organizational concept. Lack of software asset reusability standards, absence of software interoperability disciplines, and incoherent business and technology strategies drove the enterprise to establish a more suitable model that promised to foster business agility and increase return on investment. This model also galvanized the development of SOA governance best practices, introduced SOA products, and promoted new service-oriented modeling disciplines.

The enterprise is still seeking mechanisms that can alleviate alignment challenges between business and information technology (IT) organizations. This effort includes the establishment of a common service taxonomy and vocabulary—an easy-to-understand language that can fill in the communication gaps between the problem and solution domain entities and establish a proper service development life cycle.

Unlike other SOA books on the market, this one introduces a service-oriented modeling framework that employs an agile and universal business and technology language to facilitate analysis, design, and architecture initiatives. The service-oriented modeling disciplines presented will enable practitioners to integrate existing legacy applications and to incorporate new ideas and concepts to address organizational concerns. These proposed best practices can be applied to all technologies, software platforms, and languages despite their physical location or ownership. Furthermore, business and IT professionals, such as managers, business analysts, business architects, technical architects, team leaders, and developers can now share the burden of software development initiatives as they are commissioned to bear equal responsibility and accountability.

The service-oriented modeling research presented in this book was driven by the following vision statements:

- Introduce a state-of-the-art and holistic modeling language that can facilitate an SOA implementation
- Introduce advanced service life cycle concepts and processes that can be employed to manage service-oriented projects
- Enable business and IT personnel to equally partner in service-oriented modeling efforts and to represent their unique perspectives

This book's mission focuses on the following service-oriented modeling disciplines that also offer an easy-to-understand modeling language and a notation that is simple to use:

- Service-oriented conceptualization
- Service-oriented discovery and analysis
- Service-oriented business integration
- Service-oriented design
- Service-oriented conceptual architecture
- Service-oriented logical architecture

Chapter 1 introduces the proposed service-oriented modeling framework and outlines its components. This chapter targets business and technology personnel who seek to establish and implement a service-oriented modeling language that can be employed during projects to guide, monitor, and control service development life cycles.

Part One discusses the service life cycle model and its various building blocks. It elaborates on service evolution management mechanisms during given projects and business initiatives. It also discusses various life cycle perspectives that enable monitoring and assessment of a project's process. Chapters 2 and 3 set the stage for the management of service-oriented modeling disciplines that are discussed throughout the book. They provide a solid framework for the service-oriented modeling environment and will be practical for business and IT executives, product managers, project managers, architects, and lead developers.

Part Two is dedicated to the service-oriented conceptualization process and elaborates on various mechanisms that can help organizations to establish common concepts and identify conceptual services and establish enterprise taxonomies. This Part is targeted at product managers, business architects, business analysts, technical architects, technical managers, team leaders, and developers. Chapters 4 and 5 introduce a step-by-step and an easy-to-employ concept discovery process that yields conceptual solution propositions to organizational problems.

Part Three delves into service-oriented discovery and analysis mechanisms. This Part furnishes best practices and procedures that should be used to discover new services and even employ legacy applications to provide viable business and technological solutions. Chapter 6 provides unique mechanisms to establish services' identities and to categorize them based on their distinctive characteristics. Chapter 7 enables product managers, business architects, technical architects, business analysts, technical leaders, and developers to perform service-oriented analysis on identified software assets. Chapter 8 introduces an analysis proposition modeling process that employs a service-oriented analysis language that can be further used in future service design, architecture, and construction phases.

Part Four depicts service-oriented business integration mechanisms and furnishes a business modeling language that can be used to integrate services with business domains and business products. Chapters 9, 10, and 11 expand on industry-standard business architecture and propose an implementation of business architecture disciplines. Business product managers, business managers, IT managers, business architects, technical architects, business analysts, and developers will find these chapters useful for alignment initiatives between business and technology organizations.

Part Five focuses on service-oriented design strategies, service relationships, logical compositions of services, and service behavior analysis. Chapters 12, 13, and 14 target analysis, architecture, and development personnel; these individuals must understand the nature of service-oriented software relationships as well as prepare packaged solutions for the architecture and construction teams, study service behaviors, and devise service-oriented transactions.

Finally, Part Six elaborates on fundamental aspects of service-oriented software architecture. These topics include conceptual and logical architecture modeling disciplines. Chapters 15 and 16 offer a conceptual architecture modeling language that can be employed to describe organizational technological abstractions, as well as a logical architecture topic that depicts the fundamental service-oriented building blocks that will be deployed to production and become an integral part of an organization's physical architecture.

ACKNOWLEDGMENTS

For all of those who have helped shape this book by contributing their ideas, vision and strategies, many thanks. Without their involvement, enthusiasm, and the coherent direction they provided, this book would not have been possible to accomplish.

The valuable insights and feedback that were extended through the research and the actual writing process were concerned with many fundamental aspects of business and technology practices. These include the general strategy of the book, business modeling aspects, business process disciplines, business architecture policies, enterprise and application architecture standards, computer programming topics, and even editorial contributions.

Charu Bansal, Donald Buckley, Dolly Dsa, Donald Mahaya, Eric Marks, Boris Minkin, Lisa Nathan, Mark Penna, Hormazd Pochara, Monica Roman, Jeff Schneider, and Alex Rozen. To all of you thank you for your time and commitment to the success of this book.

Special thanks to Mr. Sheck Cho, the executive editor that not only encouraged this project, but also provided strategic and tactical support throughout the publishing process.

CHAPTER 1

INTRODUCTION

As human beings, we are passionate about new ideas that promise to transform our lives and create new opportunities. We also tend to rapidly replace old technologies with new ones. Ours is a versatile society that runs on tomorrow's software piled on top of the technology layers of yesterday and today.

Try to imagine the next breakthrough that will supersede today's examples of human ingenuity. Will it be miniature software installed on microwave ovens or refrigerators that monitors a diet prescribed by a personal nutritionist? Could it be a smart software component that not only designs itself but also architects its own operating production environment? Or perhaps a virtual software development platform that enables business and technology personnel to jointly build applications with goggles and gloves?

These futuristic software concepts would probably contribute yet another layer to our already complex computing environments, one requiring resources to maintain and budgets to support. This layer would sit on top of technological artifacts accumulated over the past few decades that are already difficult to manage.

Not long after the new millennium, discontent over interoperability, reusability, and other issues drove the software community to come up with the service-oriented architecture (SOA) paradigm. Even readers who are not familiar with SOA will probably agree that it is rooted in traditional software development best practices and standards. To fulfill the promise of SOA, superb governance mechanisms are necessary to break up organizational silos and maximize software asset reusability. The SOA vision also addresses the challenges of tightly coupled software and advocates an architecture that relies on the loose coupling of assets. On the financial front, it tackles budgeting and return-on-investment issues. Another feature that benefits both the technological and business communities is a reduction of time to market and business agility. Indeed, the list of advantages continues to grow.

But does the promise of SOA address software diversity issues? Does it offer solutions to the integration and collaboration hurdles created by the accumulation of generations of heterogeneous computing landscapes? Will the SOA vision constitute yet another stratum of ideas and technologies that will be buried beneath future innovations? Will SOA be remembered as a hollow buzzword that failed to solve one of the most frustrating technological issues of our time? Or will it serve as an inspiration for generations to come?

It is possible that SOA may fail to deliver on its promise, but if it does, we, business and IT personnel, must shoulder some of the blame. SOA may turn out to be little more than a technological fire drill if we are ambivalent about the roles and responsibilities of legacy software in our existing and future organizational strategies; if we fail to tie together past, present, and future software development initiatives; and if we disregard the contributions of previous generations of architectures to today's business operations. Indeed, the idea of properly bridging new and old software technologies is a novel one. But what about establishing a more holistic view of the technological inventory that we have been building up for years? Can we treat all our software

assets equally in terms of their analysis, design, and architectural value propositions? Can we understand their collaborative contribution to our environment without being too concerned about their underlying languages and implementation detail? Can we name these assets *services*? Can we conceive of them as *service-oriented* entities? Are they not built on similar SOA strategies and principles?

This book introduces service-oriented modeling mechanisms that will enable us to conceive software products that we have been constructing, acquiring, and integrating during the past few decades as *service-oriented* constituents. These entities—either legacy applications written in languages such as COBOL, PL1, Visual Basic, Java, C++, C#, or diverse empowering platforms and middleware—should all take part in an SOA modeling framework. Most important, they should be treated equally in the face of analysis, design, and architectural initiatives, and should simply be recognized as services.

A new SOA modeling language will be unveiled in this book that is *not* based on any particular programming language paradigm, constrained by language structure barriers, or limited to a language syntax. As a result of this universal language, the modeling process becomes more accessible to both the business and technology communities. This SOA modeling approach is well suited to provide tactical, short-term solutions to enterprise concerns, yet it furnishes strategic remedies to persistent organizational problems. So what is service-oriented modeling?

> Service-oriented modeling is a software development practice that employs modeling disciplines and language to provide strategic and tactical solutions to enterprise problems. This anthropomorphic modeling paradigm advocates a holistic view of the analysis, design, and architecture of all organizational software entities, conceiving them as service-oriented assets, namely services.

SERVICE-ORIENTED MODELING: WHAT IS IT ABOUT?

Modeling activities are typically embedded in the planning phase of almost any project or software development initiative that an organization conducts. The modeling paradigm embodies the analysis, design, and architectural disciplines that are being pursued during a given project. These major modeling efforts should not center only on design and architectural artifacts such as diagrams, charts, or blueprints. Indeed, modeling deliverables is a big part of a modeling process. But the service-oriented modeling venture is chiefly about simulating the real world. It is also about visualizing the final software product and envisioning the coexistence of services in an interoperable computing environment. Therefore, the service-oriented modeling paradigm advocates first creating a small replica of the "big thing" to represent its key characteristics and behavior—in other words, plan small, dream big; test small, execute big!

A VIRTUAL WORLD. How is it possible to simulate such a business and technological environment that offers solutions to organizational business and technology problems? "Simulating" does not necessarily mean starting with the construction of a software executable. It does not imply instantly embarking on an implementation initiative to produce source code and build components and services. The service-oriented modeling process begins with the construction of a miniature replica on paper. This may involve modeling teams in whiteboard analysis, design, and architecture sessions, or even the employment of software modeling tools that can visually illustrate the solutions arrived at. Thus, the simulation process entails the creation of a virtual world in which software constituents interface and collaborate to provide a viable remedy to an organizational concern.

A STRATEGIC ENDEAVOR GUIDED BY MODELING DISCIPLINES. Creating a miniature mockup of a final software product and its supporting environment can obviously reduce investment risk by ensuring the success of the impending software construction initiative. This can be achieved

by employing analysis, design, and architectural disciplines that are driven by a modeling strategy that fosters asset reusability, a high return on investment, and a persuasive value proposition for the organization.

Service-oriented modeling disciplines enable us to focus on modeling strategies rather than being concerned with source code and detailed programming algorithms. By employing this modeling paradigm we raise the bar from the granular constructs of our applications, yet we must accommodate the language requirements of the underpinning platforms. We focus on identifying high-level business and technological asset reusability and consolidation opportunities, but we must also foster the reuse of software building blocks such as components and libraries. We rigorously search for interoperability solutions that can bridge heterogeneous technological environments, but we also concentrate on integration and message exchange implementation detail.

A LEARNING AND VALIDATION PROCESS. By producing a small version of the final artifact we are also engaging in a *learning* and verification process. This activity characteristically would enable us to validate the hypothesis that we have made about a software product's capability and its ability to operate flawlessly later on in a production environment. We are also being given the opportunity to inspect key aspects of software behavior, examine the relationships between software components, and even understand their internal and external structures. We are involved in a software assessment process that validates the business and technological motivation behind the construction of our tangible services.

To better understand the key characteristics of a future software product and its environment, the assessment effort typically leads to a proof-of-concept, a smaller construction project that concludes the service-oriented modeling initiative. This small-scale software executable, if

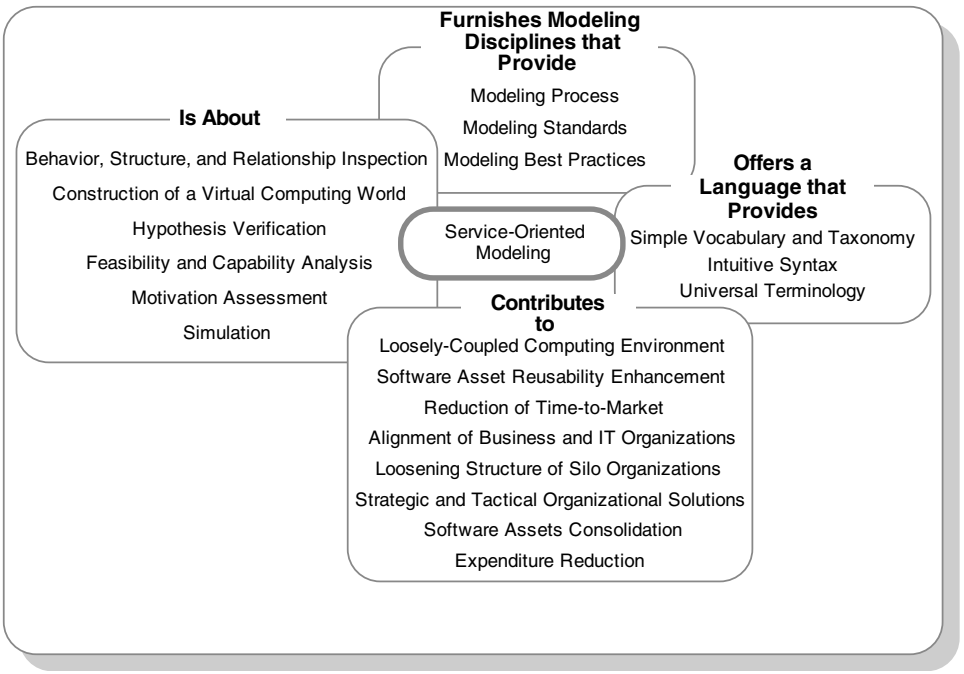


EXHIBIT 1.1 ESSENCE OF THE SERVICE-ORIENTED MODELING PARADIGM