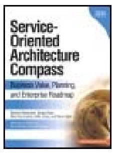


Service-Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap

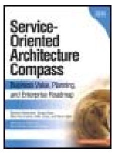
By Norbert Bieberstein,, Sanjay Bose,, Marc Fiammante,, Keith Jones,, Rawn Shah

....
.

Publisher: **IBM Press**
Pub Date: **October 19, 2005**
ISBN: **0-13-187002-5**
Pages: **272**

Overview

Maximize the business value and flexibility of your SOA deployment. In Service-Oriented Architecture Compass, IBM experts offer a complete roadmap for maximizing the business value and flexibility of SOA in your environment. Drawing on their unsurpassed experience and enterprise SOA migrations, the authors share best practices for planning, architecting, implementing, managing, and securing SOA. They clearly explain what SOA is, the opportunities it offers, and how it differs from earlier approaches. Then, using detailed examples from IBM consulting engagements, they show how to deploy SOA solutions that tightly integrate with your processes and operations. Whether you're an enterprise architect, project manager, or software development leader, Service-Oriented Architecture Compass will demonstrate that SOA will help your enterprise integrate its business processes with its technical infrastructure to deliver on goals, provide agility, and adapt to an ever-changing business world.



Service-Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap

By Norbert Bieberstein,, Sanjay Bose,, Marc Fiammante,, Keith Jones,, Rawn Shah

....
.

Publisher: **IBM Press**
Pub Date: **October 19, 2005**
ISBN: **0-13-187002-5**
Pages: **272**

Table of Contents | Index

PREV < Day Day Up > NEXT

Copyright

IBM Press Program Manager: Tara Woodman, Ellice Uffer

IBM Press Consulting Editor: Linda Foo

Cover design: IBM Corporation

Published by Pearson plc

Publishing as IBM Press

Praise for Service-Oriented Architecture Compass

"Service-Oriented Architecture enables organizations to be agile and flexible enough to adopt new business strategies and produce new services to overcome the challenges created by business dynamism today. CIOs have to consider SOA as a foundation of their Enterprise Applications Architecture primarily because it demonstrates that IT aligns to business processes and also because it positions IT as a service enabler and maximizes previous investments on business applications.

"To understand and profit from SOA, this book provides CIOs with the necessary concepts and knowledge needed to understand and adapt it into their IT organizations."

—Sabri Hamed Al-Azazi
CIO of Dubai Holding, Sabri

"I am extremely impressed by the depth and scale of this book! The title is perfect—when you know where you want to go, you need a compass to guide you there! After good IT strategy leads you to SOA, this book is the perfect vehicle that will drive you from dream to reality. We in DSK Bank will use it as our SOA bible in the ongoing project."

—Miro Vichev
CIO, DSK Bank, Bulgaria
member of OTP Group

"Service-Oriented Architecture offers a pathway to networking of intra- and inter-corporate business systems. The standards have the potential to create far more flexible and resilient business information systems than have been possible in the past. This book is a must-read for those who care about the future of business IT."

—Elizabeth Hackenson
CIO, MCI

"Service-Oriented Architecture is key to help customers become on demand businesses—a business that can quickly respond to competitive threats and be first to take advantage of marketplace opportunities. SOA Compass is a must-read for those individuals looking to bridge the gap between IT and business in order to help their enterprises become more flexible and responsive."

—Michael Liebow
Vice President, Web Services and Service-Oriented Architecture
IBM Business Consulting Services

More Praise for Service-Oriented Architecture Compass

"This book is a welcome addition to SOA literature. It articulates the business case and provides practical proven real-world advice, guidance, tips, and techniques for organizations to make the evolution from simple point-to-point web services to true SOA by addressing such topics as planning, organization, analysis and design, security, and systems management."

—Denis O'Sullivan
Fireman's Fund Enterprise Architect

"The book Service-Oriented Architecture Compass shows very clearly by means of real projects how agile business processes can be implemented using Service-Oriented Architectures. The entire development cycle from planning through implementation is presented very close to practice and the critical success factors are presented very convincingly."

—Professor Dr. Thomas Obermeier
Vice Dean of FHDW Bergisch Gladbach
Germany

"A comprehensive roadmap to Service-Oriented Architecture (SOA). SOA is, in reality, a business architecture to be used by those enterprises intending to prosper in the 21st century. Decision makers who desire that their business become flexible can jumpstart that process by adopting the best practices and rules of thumb described in SOA Compass."

—Bob Laird
MCI IT Chief Architect

"This book is a major improvement in the field. It gives a clear view and all the key points on how to really face a SOA deployment in today's organizations."

—Mario Moreno
IT Architect Leader, Generali
France

IBM Press: The developerWorks® Series

Forewords

Computing has evolved in dramatic ways since the first abstracting systems were developed in the late 1930s and early 1940s. In the ensuing decades, the usual metrics associated with computing have increased by many orders of magnitude: speed of computation, capability to store and retrieve information, capability to transmit information, programmability, and so on. The twenty-first century is witnessing yet another transformation of computing: its immersion into a networked environment. The confluence of computing and communication has produced a fertile environment for innovation and reinvention in computation. This book is about one very important example of new thinking: the so-called Service-Oriented Architecture.

The potential applicability of computers to administrative tasks was recognized very early in their development. The Hollerith punched card, invented in the 1880s by Herman Hollerith, was used in connection with the 1890 U.S. census. Hollerith based his punched cards on the earlier punched card design used in 1804 in Joseph-Marie Jacquard's eponymous loom. Although some of the earliest applications were driven by military needs (for example, ballistics and cryptanalytic computations), after World War II, computers were turned to civilian use in the conduct of business. Payroll, inventory, accounts receivable and payable, sales, production, numerical control, and a host of other business data-processing applications were developed to run on expensive mainframe machines. These so-called data-processing systems were often housed in glassed-in "fishbowls" and were exhibited proudly by their owners as evidence that the company was at the forefront of high-technology application. Programming languages such as Common Business Oriented Language (COBOL) were developed for the business world as a counterpart to Fortran's dominance in scientific and engineering computing. Of course, since that now-long-ago era, many other high-level languages and systems have been developed for business applications.

Many of these business-oriented applications and systems deal with vast databases containing billions and, in some cases, trillions of bytes of information. Modern database technology allows for hierarchical storage structures capable of manipulating and storing terabytes to petabytes of information. (A petabyte is 1015 bytes.) In most such systems, the programs that interact with the databases and with each other do so either through the sharing of access to a common data storage system or by direct exchange through application programming interfaces (APIs) on a shared computer. Distributed database systems are federations of database systems that typically maintain multiple copies of data, keeping them synchronized by a variety of pairwise and group coordination procedures. This program-centric view of data structures and file exchanges has dominated the data management scene for decades. Only as networking has become more widespread has this philosophy been given serious reconsideration.

Rather than thinking of database management (local or distributed) as a programming problem involving the programmatic updating of storage structures, a network-centric view of data management sees the interaction among the systems as a layered architecture of services. This is very much aligned with the Internet's architecture, which sees remote systems as servers or as peers in a client-server or peer-to-peer system. In this view, protocol replaces API as the primary mechanism of exchange between programs. The protocol specifies the format of exchanged information and the procedures by which the information is exchanged. The APIs used in one machine need bear no relation to those used in another. All the commonality lies in the procedures and formats for protocol-driver data exchange.

This is a profoundly different way of looking at data handling in a networked environment. Rather than the bulk transfer of an update file that is then run through a local program that executes the database update, the source(s) and sink(s) of the traffic exchange transactions in accordance with agreed protocols. Each machine expects the others to provide services in aid of database management in accordance with the agreed-upon and standardized protocols.

Peer-to-peer data and file sharing operates along these lines—all participants agree on a protocol, format, and representation for the exchanged data. That this is a powerful form of information processing is revealed by statistics that show that up to half of the Internet's capacity is taken up with such transactions.

This book takes the reader, in some careful detail, through the concepts and operational issues associated with the creation of a Service-Oriented Architecture (SOA) for distributed information processing. It is hard to imagine a more important topic in the twenty-first-century information infrastructure. Some important side effects of this reparsing of the data-processing architecture are that business IT continuity will be easier to establish, interenterprise transactions will be easier to implement, security and authenticity of the transacting parties will

Preface

Early in 2004, a small team within the IBM Enterprise Integration team was asked to draft an IBM internal SOA cookbook to document SOA engagement experiences and share best practices. By engaging subject matter experts across IBM and infusing our own project experiences, we created the first version of a SOA cookbook. This was well received by the IBM SOA technical community, and a general awareness of this book spread to our key customers. Some of these customers started requesting a formal and public version of the cookbook. This brought us together to distill the internal cookbook and author a SOA book for the general SOA community.

Trademarks and Notices

developerWorks® Link Icons Used in This Book

A.1

Margin icons are used to indicate that links to further resources related to the text are available at the developerWorks Web site on ibm.com. These links are listed at the end of each chapter and an electronic guide is available through ibm.com/developerworks/dwbooks.

Acknowledgments

To write this book, we relied on the advice, expertise, knowledge, and contribution of a number of our IBM colleagues. Most of them are actively engaged in SOA-based customer projects and IBM software product development.

We would first like to thank the key content contributors to this book. With their substantial effort and deep subject expertise, it was possible to give insights into a broad range of SOA topics. Randy Langel provided input for Chapter 2, "Explaining the Business Value of SOA." The topic of information and data services was elaborated on by Mei Selvage in Chapter 3, "Architecture Elements." Greg Flurry and Eoin Lane helped articulate the sample assets in Chapter 6, "Enterprise Solution Assets." Heather Hinton, a security architect in IBM Tivoli product development, provided content for Chapter 8, "Securing the SOA Environment." Members of the Tivoli team, including Rosalind Radcliffe, Ingo Averdunk, Sudhakar Chellam, David Cox, Steve Tremper, and John Whitfield, provided content for Chapter 9, "Managing the SOA Environment." We would also like to thank Derek Ireland of Standard Life, Dr. Martin Schneider of the Austrian Ministry of Justice, and Anton Fricko for their help with Chapter 10, "Case Studies in SOA Deployment."

Several technical experts helped review individual chapters. We appreciate their valuable feedback and input—Jonathan Adams, Yvonne Balzer, Maryann Hondo, Heather Kreger, Rick Robinson, and many others who were supportive in various subject areas and deserve our recognition.

The team from IBM Press was instrumental in thoroughly reviewing this book and providing overall guidance and feedback. We want to thank them for improving various aspects of the book. Kevin Davis and David Kane did two rounds of rigorous technical reviews and provided critical inputs. Ginny Bess Munroe provided excellent language-editing skills by streamlining the text and Amy Lepore patiently copy edited the chapters. Lori Lyons led the project in the final stages. And finally, Paul Petralia provided overall editorial leadership and liaison.

We would also like to thank Vinton Cerf, Daniel Sabbah, and Jason Weisser for generously donating their time to write the forewords for this book. The work took time from our daytime jobs, and we thank our respective management at IBM for their understanding and for granting us the necessary freedom and support.

Finally, we would like to thank our families and friends for their ample encouragement and support. Thank you for your infinite patience during the last year while this book was being prepared. It would have been impossible without your support and understanding.

About the Authors

Norbert Bieberstein is a solution architect for the IBM Enterprise Integration team and is responsible for the team's worldwide communication. In his dual role, he gained first-hand experiences from customer projects in various industries striving to migrate to SOA-based On Demand solutions. He currently is completing his MBA at Henley Management College in the United Kingdom. In his communication role, he is delivering insight and best practices to IBM and customers in various forms. Norbert co-authored the IBM Redbooks Introduction to Grid Computing with Globus (SG24-6895-01) and Enabling Applications for Grid Computing with Globus (SG24-6936-00), wrote the textbook CASE-Tools (ISBN: 3446175261), and published several magazine articles on various IT topics. Norbert also worked as a technology manager in the IBM software partner organization, where he led the IBM OMG delegation during UML definition. He also acted as a software engineering (CASE) consultant to the IBM software development labs. Norbert has more than 25 years of experience in information technology and computer sciences. Before joining IBM in 1989, he was an application developer for a regional CIM provider and worked as scientific programmer at Aachen University of Technology (RWTH), where he received his masters in mathematics and geography. He also holds teacher's degrees for higher education in Germany. He lives with his family near Düsseldorf, Germany.

Sanjay Bose is the Design Center leader for the IBM Enterprise Integration team. He has more than 12 years of IT industry experience, primarily focused on creating product architecture and design, articulating technical strategy, and designing enterprise application systems using distributed technologies. He currently leads the design center to identify IBM software portfolio requirements and to develop solution components and assets by engaging enterprise clients and IBM software product development laboratories. His areas of expertise include service-oriented architecture, enterprise service bus, Web services, J2EE, and e-business technologies. Sanjay also worked in product development on the WebSphere Application Server and the WebSphere Portal Server. He has published several technical papers and also has contributed to industry specifications and standards. Sanjay received his bachelor's degree in computer science and engineering from the Indian Institute of Technology (IIT) in Mumbai, India and has completed MBA coursework at the Tepper School of Business, Carnegie Mellon University. He lives and works in Pittsburgh, Pennsylvania.

Marc Fiammante is an IBM Distinguished Engineer, elected to the IBM Academy of Technology in 2003, with wide experience in large project architecture and software development on multiple environments. He is the chief architect of the European, Middle East, Africa, and Asia-Pacific Enterprise Integration Solutions team. Marc has 21 years of experience in IT. He has filed several software domain patents and has published several articles related to e-business technologies. He leads architecture teams in major industry projects. He has architectural and technical expertise with service-oriented architecture, Web services, enterprise application integration, and e-business and object-oriented technologies, including a number of software middleware systems, programming languages, and standards. Marc is a graduate engineer of the Ecole Centrale de Paris.

Keith Jones, PhD, is currently a leading IT architect at IBM Enterprise Integration Solutions, where he focuses on the definition and implementation of service-oriented architectures with leading-edge customers. He has 30 years of experience in the IT industry as a systems engineer, software developer, strategist, systems architect, and author of many middleware publications. Keith's professional interests center on building transactional, message-oriented, and service-oriented middleware infrastructures in support of business processes in a wide range of customer environments. Most recently, these have included infrastructures at major financial services, retail services, automotive manufacturing, online media, and auction enterprises. Keith has a PhD in chemistry and lives with his family in Boulder, Colorado.

Rawn Shah is the Community Editor (and, formerly, the SOA and Web services Zone Editor) for IBM developerWorks. Rawn has 12 years of experience in the IT industry, serving in various roles including positions as a network administrator, an application developer, a vice president of a regional Internet service provider, a columnist, an author, and an editor. He has written more than 280 articles for dozens of technology magazines, including CNN.com, NetworkWorld, JavaWorld, NC World, Windows TechEdge, and LinuxWorld, and he was directly involved in the release of the industry-leading publications JavaWorld and LinuxWorld in the mid-1990s. His interests lie in finding new ways for facilitating the communication and collaboration of technical ideas and processes between distributed audiences and transferring this knowledge in meaningful ways to nontechnical audiences such as business teams. He and his family currently reside in Tucson, Arizona.

developerWorks and SOA

Through the SOA and Web services Zone on developerWorks, IBM helps software developers and architects by providing them with technical content, tools, and resources that enable them to build on demand applications in an open environment. Our goal is to provide new, original content that guides developers both in their thinking and implementation of Web services creation and development of a Service-Oriented Architecture.

A sampling of content includes topics that are about both cutting-edge developments as well as some of the basics:

- Business processes— What are the roles of Web services and SOA in creating and managing business processes? How is this particularly relevant for those architecting on demand solutions?
- Migration— Is it possible to migrate to a SOA? What are the steps I take to do that?
- Model-Driven Architecture— How does the Rational portfolio enable more efficient creation and modeling of services?
- Standards— I'm aware of Web services standards, but not sure how to address them in my development process.
- The Enterprise Service Bus— Why do I need it?

These comprise just a small overview of the categories that we touch on in the SOA and Web services Zone on developerWorks. We are also aware that some are looking to understand methodology and philosophy of creating certain kinds of systems, and others want practical advice on delivery and implementation. Every month delivers a number of articles, tutorials, downloads, sample code, and community resources that address the broad array of issues related to services development.

Because of IBM's position as the leader in open, cross-platform software development, developerWorks benefits from having access to some of the most accomplished thinkers in the SOA and Web services space. Some of these people are regular contributors through their article series, blogs, and participation in our online forums, Technical Briefings, and Web-based seminars.

We invite you to look around our zone and see for yourself how it might help you with your development and architecture goals:

ibm.com/developerWorks/webservices

Chapter 1. Introducing SOA

"Once an organization loses its spirit of pioneering and rests on its early work, its progress stops."

—Thomas J. Watson, Sr.

The increasing pace of the evolution of business requirements and the need for increased revenues and cost optimization are leading corporate executives to deliberately align their IT organizations more closely with their business requirements. The main goal of this convergence is to develop more optimal and operationally integrated business processes that can be implemented by departments, business units, and business partner networks. Historically, this convergence process has been constrained because IT systems have not kept in step with business needs and because the IT infrastructure has inherent operational and developmental limitations, such as proprietary programming interfaces that restrict a system's flexibility. Consequently, the integration challenge demands a technology that can successfully bring together the needs of business and IT into a viable solution that not only makes efficient and effective use of the IT infrastructure, but that is also flexible and adaptive enough to keep pace with continual changes in the organization's business processes and business models.

Integration challenges are often considered to be only a technical interoperability issue. A change to a business need supported by functions that an application provides is considered to be a change to the actual software program, how it is deployed, or how it interacts with other applications. When a new business need arises, organizations order the revitalization, creation, or installation of new applications. However, this view of the integration challenge assumes a prevailing and potentially long-term view of any application, limiting the overall flexibility and agility of the business. Therefore, we need to reexamine the approach to and the source of this challenge to unlock a new way to address the issue of business-driven software changes and implementation.

The prevailing integration challenge in the software industry stems from the very same force that has been the energy of this industry: the diversity of methods and approaches deployed to solve a multitude of essentially technical issues. This positive force has allowed software engineers to tackle increasingly complex problems with new ideas, new methods, and new technologies. It has also helped to speed the growth of most, if not all, other industries with the rise of computerization by exploiting a range of hardware solutions, operating environments, application systems, programming languages, and software development techniques.

Yet it is this same evolving diversity that has allowed the creation and multiplication of barriers among the many solutions that have been developed. The issue of incompatibility among solutions has become a paramount problem, not just because of the technical issues that must be resolved but because it introduces barriers to the way companies use IT to do business and interact with others.

As businesses invest in more complex and more richly featured software technologies, they also separate the world of technical operations from that of business operations. However, focusing only on technical aspects and not considering the organizational or business issues often leads to an information and process-integration failure.

Integration is an old story. In 1967, Paul Lawrence, the Louis Kirstein Professor of Human Relations at Harvard Business School, together with Jay Lorsch, addressed the organizational issues related to departmental efficiency and enterprise integration and defined integration as the "state of collaboration that exists among departments that are required to achieve unity of effort."

Until now, and as noted previously, most solutions for aligning technology and business requirements have been too technology oriented and application focused. Recently, enterprise application integration (EAI) technology has been viewed as the solution to the business-IT alignment problem. EAI, although it initially solved significant coordination problems, has failed to evolve to address the current complex array of integration issues, and it has often failed to deliver the expected business flexibility.

In addition to a focus on technology-derived solutions, there is the problem of language. There is a clear division between the language used by the business operations community and the technical jargon of the IT community. This makes it increasingly difficult to relate IT operations to business management and to relate technical problems to business issues. Interestingly enough, one of the most common problems of implementing software is not the complexity of creating a technology solution; instead, it is the problem of developing an

1.1. SOA to the Rescue

The concept of service-oriented architecture (SOA) offers a framework for better-integrated systems that meet business needs. This concept is a new rationalization of practices and techniques that already exist in process-driven, top-down, bottom-up, and meet-in-the-middle methods of software integration. It has now come to reality with the evolution of technology and the advent of true interoperability.

SOA envisions the implementation of a services platform consisting of many services that signify elements of business processes that can be combined and recombined into different solutions and scenarios, as determined by business needs. This capability to integrate and recombine services is what provides the closer relationship between business and IT, as well as the flexibility to address new situations.

The role of the SOA services platform is to provide a foundation for delivering essential business services in a flexible, easily composed, and highly reusable fashion. It is essential that an enterprise adopts a global ecosystem approach to building such a platform by providing focus on the lifecycle for business-driven IT services as they are created, evolved, and phased out during the lifetime of the organization and on using a consistent architectural approach to providing flexible and reusable business service delivery.

The need for an enterprise ecosystem view has led to the creation of the service-oriented architecture, through which composite applications can be created, modified, and removed dynamically using services, abstracted from existing applications and data, provided by the platform, or provisioned from external sources.

From a business point of view, an SOA can be expressed as a set of flexible services and processes that a business wants to expose to its customers, partners, or internally to other parts of the organization. In this context, these same services can be recombined and supplemented to support changes to or an evolution of business requirements and models over time. As an example, an auto parts supplier that uses an SOA can expand to support new automobile brands and can integrate new parts catalogs without impacting its existing business processes.

From the technical point of view, SOA defines software in terms of discrete services, which are implemented using components that can be called upon to perform a specified operation for a specific business task. The SOA concept evolves the existing software concept of a function—a specific piece of code that performs one particular task—to include the notion of a contract, a technology-neutral but business-specific representation of the function. In the auto parts supplier example, a common service might be the "catalog service" that exposes catalogs from different providers in a common way throughout the enterprise.

The concepts of service-oriented software and architectures based on such software have existed in some form for a number of years. We have found service-oriented implementations in COBOL/CICS® mainframe systems, object-oriented C++ systems, and in Java® application platforms. Some of these concepts have evolved from older ideas and from a variety of software technology origins. Thus, with no surprise from an industry that thrives on the diversity of technical ideas, the definition of what constitutes an "SOA" varies widely across the industry.

Before we create a working definition of SOA to be used throughout this book, we must understand the different ways that SOA has commonly been defined and clarify our view of what "SOA" really means.

1.2. Exploring SOA

There are a number of ways to look at the concept of SOA. In its most basic form, SOA is an approach to building software systems that is focused on how the software is actually implemented. To understand this approach and implementation directive, we need to look at individual parts of the definition and the dimensions of a typical enterprise SOA.

1.2.1. The Term "SOA"

The following are common examples of SOA definitions. Most of these definitions focus on the technical aspects of architecture, although some include business characteristics. These definitions, which we gathered from multiple sources, are interesting because they illustrate the number of different views or expressions of what an SOA is; however, they all tend to coalesce into a common meaning:

- A business definition: A set of business, process, organizational, governance, and technical methods to reduce or eliminate frustrations with IT and to quantifiably measure the business value of IT while creating an agile business environment for competitive advantage.

- Another business definition (introduced by IBM): A service-oriented architecture provides the flexibility to treat elements of business processes and the underlying IT infrastructure as secure, standardized components (services) that can be reused and combined to address changing business priorities.[1]

  [1] In an informal survey of some of its business customers done by IBM in late 2004, the previous definition received an overwhelming 90 percent support from business executives of major industries.

- The widest technical (and rather minimalist) definition: An enterprise-wide IT architecture that promotes loose coupling, reuse, and interoperability between systems.

- A moderately complex technical definition: An application architecture in which all functions or services are defined using a description language and have callable interfaces that are called to perform business processes. Each interaction is independent of each and every other interaction and the interconnect protocols of the communicating devices. Because interfaces are platform independent, a client can use the service from any device using any operating system in any language.

- The least common denominator definition: A system architecture in which application functions are built as components (services) that are loosely coupled and well-defined to support interoperability and to improve flexibility and reuse.

- The narrowest definition: SOA is a synonym for solution architectures making use of Web service technologies such as SOAP, WSDL, and UDDI. Here SOA is defined as "any product and project architecture conforming to the W3C Web services architecture (WSA)."

In this book, we elaborate on the principles of these definitions, particularly the first four, and we do not restrict ourselves to the final, narrower definition. For our discussion, we have chosen to adapt the definition of SOA that resulted from a survey of business executives:

A service-oriented architecture is a framework for integrating business processes and supporting IT infrastructure as secure, standardized components—services—that can be reused and combined to address changing business priorities.

In our view, there can be valid SOAs that do not define a single Web service and instead use existing technology such as mainframe transactional or object-oriented systems. Similarly, there are valuable Web

1.3. A Preview of the Service-Oriented Architecture Compass

SOA can fundamentally change the direction taken by an enterprise to development and deployment of business software systems for competitive advantage. Introducing SOA at the enterprise level cannot reasonably be reduced to a single project because it will affect the way that IT responds to the business requirements. When an enterprise starts on its journey to SOA, a compass is an essential navigational instrument for finding the right direction. Therefore, any book on SOA that gives concrete prescriptions for business componentization and technology implementation must be based on a common understanding of SOA—the concepts, recommendations, and best practices.

But to say that service-oriented architectures are here to solve all of our technical and business problems would be, of course, a gross exaggeration. There are numerous other problems that introduce hurdles for interoperability, compatibility, and well-mannered operation among diverse systems. Furthermore, like any other technological system, SOA deployment requires careful business-case evaluation, planning, design, implementation, and management by a properly skilled and talented team to become a successful endeavor.

This book is about making SOA real in terms of justifying action to gain business value, planning for adoption, and developing good strategies for implementation. It also proposes that even if we assume that there is consensus on methodology within a particular organization, a discussion on the nature of business and technical services within an enterprise is needed. This can help avoid misperceptions and misunderstandings, especially when extended to include a wider audience such as business partners.

Building the business case for a service-oriented architecture is not a simple task. The return on investment, as with all reuse strategies, might not be immediate. Of course, business cases are specific to each environment, and Chapter 2 discusses some of the elements that we have found to be valuable in our experience.

SOA has brought some important business and technical concepts into the spotlight. In particular, the significance of coupling software systems and their degrees of isolation is explored in Chapter 3, "Architecture Elements." We also explore implementation alternatives. These identify the programming models, constructs, patterns, protocols, and even the programming language that will be used in a SOA implementation project.

One of the most critical decisions is to avoid placing the SOA project solely in the hands of an IT department and to foster collaboration with line-of-business representatives to get broad support across the enterprise. We share some ideas for SOA team roles and tasks in Chapter 4, "SOA Project Planning Aspects."

It is worth mentioning that conceptual architectures and frameworks defined by corporate IT might sometimes have a dubious reputation at the business level in many companies; this reputation might need to be counterbalanced when an SOA vision is presented. A common statement heard in our experience is, "We have tried this before. Why do you think it will succeed this time? How do we plan for it this time around?" Therefore, the roadmap to introduce SOA into an enterprise and the governance to ensure true reusability and control cannot be overemphasized. We dedicate an important part of Chapter 4 to these considerations.

In Chapter 5, "Aspects of Analysis and Design," we reflect on the important ideas that carry forward from previous technological eras, and new techniques that apply as business requirements are analyzed and new services identified, specified, and realized. We also discuss the role of modeling for service-oriented systems as they are planned, prototyped, and provisioned.

For effectively leveraging SOA, architectural decisions and solution patterns need to be captured and reused across all enterprise projects for maximum impact. We provide guidance on how to document them in Chapter 6, "Enterprise Solution Assets," and share some sample assets.

Our previous discussion about reach and range for service-oriented architectures was intended to direct attention to a number of non-functional requirements found in SOA deployments. Those aspects are discussed in Chapter 7, "Determining Non-Functional Requirements." The concept of SOA does not deliver all necessary means to tackle these issues, but rather, SOA should be regarded within a broader context of an operating environment for on-demand applications, the general blueprint for modern IT.

One important set of non-functional requirements deserves special consideration: the security risks that services can be exposed to and the malicious acts that an enterprise must defend itself against. Security under

1.4. Summary

This chapter introduced several popular definitions for SOA. We then categorized and clarified them by identifying which characteristics are desirable.

We then discussed the SOA dimensions of business value, reach and range, maturity and adoption strategies, and we illustrated how each of these impacts the work needed for an SOA deployment. Finally, we gave a working list of topics that should be considered by enterprises navigating toward SOA and shared how we elaborate these topics in this book.

Now that you have a basic understanding of what SOA is, let us look at the reasons why one would leverage it. The next chapter covers the value of building an enterprise ecosystem with an SOA approach based on both technical and business justification.

1.5. References

Crawford, C. H. Toward an on demand service-oriented architecture. IBM Systems Journal, Vol. 44, 1-2005. http://www.research.ibm.com/journal/sj44-1.html.

Galbraith, Jay R. Competing with Flexible Lateral Organizations, 2nd Edition. Addison Wesley, 1994.

Handy, Charles. The Age of Unreason. Random House, 1995.

IBM Business Consulting Services. The Global CEO Study 2004. IBM, 2004.

Joseph, J., Ernest, M., and Fellenstein, C. Evolution of grid computing architecture and grid adoption models. IBM Systems Journal, Vol. 43, 4-2004. http://www.research.ibm.com/journal/sj43-4.html.

Lawrence, Paul R. and Lorsch, Jay W. Organization and Environment: Managing Differentiation and Integration. Harvard Business School Classics, revised edition, 1986. http://harvardbusinessonline.hbsp.harvard.edu/b01/en/common/item_detail.jhtml?id=1295.

Lorsch, Jay W. and Lawrence, Paul R. Organization Planning. RD Irwin, 1972.

Chapter 2. Explaining the Business Value of SOA

"Small opportunities are often the beginning of great enterprises."

—Demosthenes

Why do some companies survive and others fail?

Corporations are built on the assumption of continuity; they focus on operations. On the other hand, capital markets are built on the assumption of discontinuity; their focus is on creation and destruction. The market encourages rapid and extensive creation and hence greater wealth building. The market is less tolerant than the corporation when underperforming over the long term. Some of the key reasons for failure are ignoring higher-value markets, the inability to address more technologically advanced competition, or competition from lower-cost sources.

Consider the following: In 1917, U.S. Steel led the Forbes list and employed 268,000 people. Now it is down to about 21,000 employees. In the same year, there were eight other steelmakers on the top 100 list and 33 other companies in the business of extracting things out of the earth. About 45% of the list was made up of such resource producers. Today, 61 of these companies no longer exist (a 61% extinction rate).

In 1987, U.S. Steel was still on the list of the 100 most valuable companies, but by that year it was in the beginning of a gradual decline. One by one, over a period of many years, the other natural resource companies slipped off the list. Although it might be slow, change is powerful.

Examination of the S&P 500 presents a similar story. Of the 500 original companies of the S&P 500 in 1957, only 74 remained on the list through 1997, an 85% reduction rate. It is interesting to note that such driving forces (see Michael Porter[1] under "References" section) still exist today, but the rate of change has increased. In other words, companies are going out of business at a faster rate today than they were in 1957, as correlated by Richard Foster and Sarah Kaplan in Creative Destruction (Currency Press, 2001). Foster and Kaplan's book indicates that technological and public-policy shifts have consistently destabilized the market, and it is occurring at an increasing rate.

[1] Professor Michael Porter of Harvard Business School's Institute for Strategy and Competitiveness numerous fundamental concepts in competition theory, which have since become well accepted knowledge and practice for about 20 years now.

Despite the dotcom bust and major bankruptcies in the United States, this phenomenon occurs worldwide. For example, the current leading steel manufacturer in Germany, Thyssen, has replaced older, well-known names like Krupp and Hoesch, which disappeared in the last 20 years. This rapid rate of extinction leads to the three core questions that will be discussed in detail in this chapter:

1. How do companies survive over the long term?
2. How can companies effectively execute a business plan that incorporates continuous business transformation?
3. How can companies achieve the greatest possible business agility?

Today, enterprises must be more dynamic than ever to survive. They need new, evolved ways of handling the competition, and their IT infrastructures must support them as they face unique challenges they didn't have to face years ago. We believe SOA is the way that companies can develop IT infrastructures capable of supporting dynamic enterprises. However, to understand SOA, you first need to understand the particular forces of change that now affect businesses.

2.1. The Forces of Change

Most companies today are pressured by customers and shareholders to drive up growth by increasing productivity and squeezing costs out of the operation. However, it can be difficult to maximize efficiency if a company is wedded to rigid, expensive, or proprietary IT systems. It may be very efficient with a proprietary system, which might be optimal if that is all the market requires. However, a rigid and proprietary system has disadvantages in terms of effectiveness, which is what is asked for from an enterprise operating in a quickly changing market. In fact, these days agility is the most valuable thing a company can buy itself as an organization—the flexibility to meet new market demands and to seize opportunities before they are lost or before the competition gets there first.

To increase flexibility, companies need to look at business operations as a collection of interconnected functions—discrete processes and services (checking customer credit information, authenticating users, and so on). Company leadership then needs to decide which of these functions are core to the identity of the business or provide aspects to differentiate the business from its competitors. The nondifferentiating aspects can be streamlined or even outsourced to a partner. If a company can mix and match the differentiated functions on demand, or even on-the-fly, in response to changing business conditions, it can have a tremendous competitive advantage in the marketplace.

This is a powerful idea in itself, but to achieve this degree of flexibility in business operations, companies need an equally flexible IT environment. In our view, this indicates the need for an IT environment that can change on demand to the needs of the business. Service-oriented architecture provides a valuable response to this need for flexibility in business operations by providing the core structure of an on-demand operating IT environment.

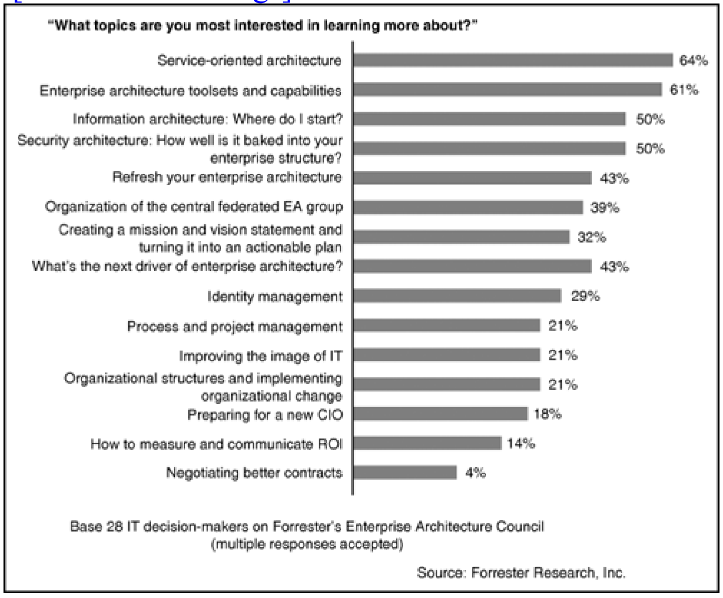
Proof of this is reflected in findings from recent surveys such as those published by Forrester Research[2] in January 2004, one of which is shown in Figure 2.1. SOA appears on top of the list in Figure 2.1. Furthermore, recent publications by Gartner Group support its growing importance and predict a predominance of service-oriented solutions like SOA and service-oriented business applications (SOBA). Charles Abrams, director of Gartner Research recently stated, "In 2001, we told you that Web services would be where they are now. Now we're telling you that they aren't done yet. Expect their impact to extend across new architectures beyond SOA, all vendor products, and new business models."[3]

[2] A survey by Forrester Research to its members of the Enterprise Architecture Council of the Forrester Oval Program, January 2004.

[3] On the Gartner event in Athens, Greece, March 15, 2005.

Figure 2.1. Forrester Research, January 2004 survey.

[View full size image]



2.2. Common Questions About SOA

First let's get to the most basic questions about SOA. To help a business executive understand the value of changing the enterprise's IT infrastructure into a service-oriented architecture, the following questions deserve convincing responses:

1. What is SOA?
2. Why do companies need SOA?
3. What benefits will businesses receive if they implement SOA?
4. What opportunities will companies miss if they don't implement SOA?
5. What is different with SOA compared to previous approaches?

The following sections attempt to provide answers to these questions. We don't provide a prescriptive method; instead, we attempt to highlight the current perspectives from both the business and IT viewpoints. Further, in this book we look in more detail at the conditions for an SOA project (see Chapter 4, "SOA Project Planning Aspects"), and we consider a more formal approach to the modeling of services and common semantics (Chapter 5, "Aspects of Analysis and Design").

2.2.1. What Is SOA?

Chapter 1, "Introducing SOA," adequately provided a technical as well as a business definition of service-oriented architecture. In summary, from the business viewpoint, SOA is a set of business, process, organizational, governance, and technical methods to enable an agile, business-driven IT environment for greater competitive advantage. It provides the flexibility to treat business processes as well as the underlying IT infrastructure as components that can be reused and recombined to address changing business priorities. Thus, in essence, SOA is the map that guides you down the road to competitive advantage.

2.2.2. Why Do Companies Need SOA?

The combination of SOA and Web services is in some ways being marketed as the silver bullet that companies have been looking for to magically solve all business issues of today. There are valid doubts about such claims, which certainly never will come true. Many business issues are not just solved by a specific IT architecture or a certain approach to making business decisions. As long as people are involved, errors are still likely to occur. However, with a foundational architecture like SOA, you can expect to do the following:

- Realize the long-promised potential of utilizing IT to extensively accelerate or improve business
- Justify IT expenses and capital outlays
- Provide nontechnical people with a clear understanding of what IT does, how they do it, and their intrinsic value

Projects that study the possibility for using SOA and its related technologies and methods (see Chapter 5) are of great help to do the following:

-

2.3. SOA Value Roadmap

This section builds a roadmap for how to reach the expected benefits and value of SOA. It describes an approach for how to explain the business value of SOA in nontechnical terms. Superiors in management will likely ask architects to summarize the values of SOA to them.

We also provide a checklist that shows the business issues you need to cover along with suggested approaches for explaining each issue. The checklist lays out the logical business arguments that will paint a full picture of the value of SOA. It starts by examining the business fundamentals of architecture, and it then works through process issues and services (or incremental delivery). This education yields an SOA business value package.

2.3.1. Explaining SOA to Business People

One of the keys to flexible business processes depends on close interaction between the IT department and the business units. This partnership is an absolute necessity for a successful SOA implementation. Currently, in most organizations, after a new business need is identified, the business operations team usually contracts the IT department to build an appropriate application that satisfies the need. Business management does not really care to know the details of how the application is built.

Similarly, the IT department builds the application only to the specifications provided and does not really look too much further into how this impacts the business. Essentially, both sides stay within their own domains, with a wall over which the business operations group throws a project to the IT group, and IT responds with a technical solution. In general, this builds too shallow of a view and understanding across the groups and, therefore, too shallow of a collaborative development process.

The IT industry has long talked about alignment and partnership with the long-term and in-depth needs of the business side, but with such limited collaborations, IT departments have been largely unsuccessful in having strategic impact on their business counterparts. Therefore, it is imperative to find a way to get the two groups cooperating in deeper relationships with each other so that they can better respect and understand the needs of the other.

Hence, educating people on the many business benefits inherent in an SOA is the first step in solidifying an active working covenant between IT and business groups. This relationship is, after all, the foundation from which all else in SOA can be built. As we show in Chapter 5, there are distinct layers of services abstractions from enterprise business components to application components. Keeping the relationships between the layers means providing a base for communication between the various levels on the business and IT sides. Chapter 4 contains a more detailed discussion about the roles involved in changing business and IT toward an SOA.

2.3.2. A Checklist for Business Change Agility

The checklist in this section shows you how to develop a business understanding of SOA. These seven steps should help you explain the technical, operational, and functional underpinnings of SOA, which are explained in greater detail in following subsections:

1. First unravel the concept of architecture from its perceived technical underpinnings and explain the business realities of this necessary business function (see Section 2.3.2.1).
2. Clarify the architect's role in providing architectural function crucial to business success (see Section 2.3.2.2).
3. Explain why it is better to realign IT elements around service definitions or business processes (see Section 2.3.2.3).
4. Emphasize the need to build an in-depth IT-to-business group dialogue that leads to a meaningful, constructive, respectful, and regularly engaged relationship between IT and the company's business units

2.4. The Nine Business Rules of Thumb for SOAs

The knowledge from the SOA business value roadmap can be condensed into the following rules of thumb that we have learned from various projects across many industries:

Rule of Thumb

Rule of Thumb 1: SOA Benefits
There are many business and technical benefits to an SOA, but none is as important as the capability for a company to respond quickly and effectively to business change and to leverage that change to gain a competitive advantage.

Rule of Thumb 2: IT and Business as Peers
You cannot build a successful SOA model if you cannot forge peer working relationships between the IT and business groups.

Rule of Thumb 3: Incremental Business Services
In an agile business, incremental business services that mirror business process steps become the core deliverables of the IT group.

Rule of Thumb 4: Business-Smart IT Architects
Business-aware IT architects are the bridge between the company's IT and business units.

Rule of Thumb 5: Opportunities for Services
Within a business process, each interaction with an IT asset is a potential location for a service.

Rule of Thumb 6: Measuring Services
A service that mirrors and executes a business process can be used to allocate IT costs and provide IT justification by correlating the IT costs with business process results.

Rule of Thumb 7: Service-Oriented Means in the Core
Companies committed to SOA will find business processes and services at the center of both business design and IT delivery.

Rule of Thumb 8: Proving Business Value of SOA
A company's SOA gives IT a definitive way to prove business value through business results measurements.

Rule of Thumb 9: Competitive Business Agility
When a change in business process no longer requires a change to application programming logic (that is, when you have a successful SOA), your company has attained competitive business agility.

2.5. Summary

As we have shown, the forces of change, enterprise reconstruction, and industry deconstruction have led to a trend toward business componentization and the use of business services. This trend requires organizations to develop agility to change their businesses as needed by their current environment. The implications and benefits of implementing SOAs have led organizations to rethink the business model toward creating reusable components that link business processes to technical services and applications.

The use of SOA is strongly dependent on a true partnership between the IT and business units of an organization in order to achieve business-change agility. As indicated in our checklist, there are seven decision and operational steps toward achieving this agility, finally leading to our rules of thumb in Section 2.4. Without an SOA, the larger concept of building an agile, on-demand enterprise is not feasible. With these business decisions in mind, we will follow with the technical concepts for service-oriented architectures in an on-demand environment that an IT organization needs to understand and take to heart.

2.6. References

Arsanjani, A. Empowering the business analyst for on demand computing. IBM Systems Journal, Vol. 44, 1-2005. http://www.research.ibm.com/journal/sj44-1.html.

Bloomberg, J. Growing an Agile Service-Oriented Architecture. ZapThink, September 2003.

Bloomberg, J. When Not to Use an SOA. ZapThink, April 2005. http://www.zapthink.com/report.html?id=ZAPFLASH-02162004.

Cecere, M. IT Trends 2004: Organizational Design. Forrester Research, February 2004.

David, Fred R. Strategic Managemen—Concepts and Cases, 7th Edition. Prentice-Hall, 1999.

Foster, Richard and Kaplan, Sarah. Creative Destruction: Why Companies That Are Built to Last Underperform the Market—And How to Successfully Transform Them, Random House, 2001.

Galbraith, Jay R. Designing the Global Corporation. Jossey-Bass, 2000.

Gilpin, M. Managing the Business Service Model. Forrester Research, April 2004.

Haeckel, S. H. Leading on demand businesses—Executives as architects. IBM Systems Journal, Vol. 42, 3-2003. http://www.research.ibm.com/journal/sj42-3.html.

Kotter, John P. The Heart of Change. Havard Business School Press, 2002.

Langel, R. Business Value of SOA. IBM Whitepaper, 2004. http://www-306.ibm.com/software/solutions/webservices/eis/businessvaluesoa.html.

Leganza, G. Managing Emerging Technology: Pearls from the EA Council. Forrester Research, March 2004. Forrester Oval Program: http://www.forrester.com/Oval/Index.

Marks, Eric A. and Werrell, Mark J. Executive Guide to Web Services, Wiley, 2003.

Moore, Geoffrey A. Inside the Tornado. Harper Perennial, New York, 1999.

Porter, Michael. Competitive Advantage: Creating and Sustaining Superior Performance, Free Press, 1985.

Porter, Michael. Competitive Strategy: Techniques for Analyzing Industries and Competitors, Free Press, 1980

Rutledge, K. (ed.). The Business Case for e-business. IBM Press, 2005.

Shi, D. and Daniels, R. I. A survey of manufacturing flexibility: Implications for e-business flexibility. IBM Systems Journal, Vol. 42, 3-2003. http://www.research.ibm.com/journal/sj42-3.html.

Taylor, B., Stiles, P., and Tampoe, M. "Governance and Performance: The Future for the Board." Strategic Dynamics, Henley Management College, 2001.

Chapter 3. Architecture Elements

"Architecture is the learned game, correct and magnificent, of forms assembled in the light."

—Le Corbusier

The main purpose of a service-oriented architecture (SOA) is to offer synergy between the business and IT groups in an organization and to offer the organization greater flexibility, as described in the preceding chapter. This chapter provides more detailed architectural perspectives and models for SOA architects and implementers.

As the title of this chapter implies, architects must examine different elements of an SOA design. The first aspect is the impact of the characteristics of an SOA—the set of implications of service componentization, the reuse of those components from variable requesters, and the capability to compose those services in business processes. The second aspect involves the various architecture domains where services can exist and the functions they provide. The characteristics define how the domains interact with each other.

Figure 3.1 presents a map that includes both aspects of SOA. In this map, the first aspect influences the relationships between the domains, as indicated by the arrows in the diagram. The domains in the rounded rectangles are, of course, the second aspect. These characteristics, which we cover in depth in Section 3.1, include how those relationships are influenced by the following factors detailed in the same section:

- Platform
- Location
- Protocols
- Programming language
- Invocation patterns
- Security
- Service versioning
- Service model
- Information model
- Data format

Figure 3.1. The domains of SOA.

[View full size image]

Enterprise Services
Business Services
Application Services
Programming Space
Off the Shelf Space

3.1. Refining SOA Characteristics

Creating reusable service components is just like building the components for a hi-fi stereo system to play music. The music components industry has progressed to a point of near ubiquity. Without knowing the technology inside each component, you can safely buy and plug together a set of components that comes from different technological concepts, manufacturers, resellers, retail outlets, and even regions of the world. To further illustrate this point, you can even take a more modern component such as a Super Audio CD player, plug it into a decades-old amplifier, and still make it work. This type of loose coupling, reusability, and flexibility is made possible by the support for common standards for both signal technical characteristics and content.

Similarly, in an SOA environment, the service component providers might not always know exactly what form the requesters will take at the time a service is created. The requesters, in turn, should not have to care about the technology behind the service implementation.

The degree of flexibility and reusability of the SOA will depend on the enterprise business model and context in use. This will drive the degree of coupling or decoupling that the architecture will need to implement.

The following characteristics address the degrees of flexibility that will affect your architecture decisions and the variability of the system you create.

3.1.1. Platform

The platform used to support a service implementation should not be relevant to consumers. This includes the intermediary layers of the operating system, the communication protocol, and perhaps even application layers. If two systems interact through interoperable protocols, such as SOAP, HTTP, or messaging middleware, each side usually does not need to consider the hardware, operating system, or server platform supporting each. Either side is free to change some or all of these aspects without really affecting the other.

However, if your target architecture for a given business model has the requesters' and providers' platforms under your direct control, you may be able to gain significant advantages in performance and system management by not separating this aspect of the architecture.

3.1.2. Location

Location is usually one of the factors leading to a more loosely coupled architecture, as several instances of the same service may exist in different locations. It is easier to reach service delivery scalability by locating multiple instances of the same service on different nodes. The identity of a service provider can be negotiated through a third-party broker component. The broker might even use geographical location, client identify, membership scheme information, transaction value, or a number of other criteria to match the service requester with a suitable service provider.

As an example, in one of our SOA projects for the defense industry, a device carried by a soldier that invokes a service from a command and control center, such as "identify target as friend or foe," will almost certainly need location independence. Depending on your environment, this aspect might imply that the infrastructure will need to perform service localization and routing appropriately.

3.1.3. Protocols

As with platform considerations, protocol independence is an aspect of SOA that may or may not be necessary depending on the business model and context. As an example, in a retail enterprise, the protocols between corporate and branches are under full control of the enterprise and can thus be unified to a common choice of protocol. Therefore, you can have a preferred binding, such as an EJB binding, for service requests that flow between the branch infrastructure and the corporate infrastructure to improve performance. This does not limit the service exposure that is available using other protocols at the same time, such as SOAP over HTTP.

Communication protocols can be defined by a configuration statement declared in an SOA service interface. In practice, this requires that you create applications using a protocol-independent service API such as JAX-RPC. In this case, the protocol binding in the service interface definition can be changed readily when necessary. For example, you can change the communications protocol from SOAP over HTTP to SOAP over Java Messaging

3.2. Infrastructure Services

As previously stated, SOA is not just a new way of writing code; it affects all elements and services of an enterprise and possibly even beyond to its network of partners. Our domain model for SOA depicts a set of infrastructure services that are the foundation for services operation. As Web service standards define the description, location, invocation, and data transport formats between services, the infrastructure services provide the execution platform and associated utility services. In a way, you can consider this domain as a virtual, distributed, operating system layer for services.

3.2.1. Resource Virtualization Services

The closest to the hardware, especially the network hardware, are the resource virtualization services. These are solutions and services that enable a platform-independent environment for the execution of services in a network. Thus, applications as services are no longer strictly bound to a specific predefined operating system or hardware platform. Services are executed in a virtual operating system that manages the available and suitable selection of servers as well the storage systems. As we show in more detail in Section 3.4.6, information integration plays an important role in enterprise and service-oriented solutions. Specific services such as hard disk space or CPU allocation fall into this "resource virtualization" category. For virtual server and storage, the IBM Virtualization Engine is a comprehensive portfolio of systems technologies and tools that can help you aggregate pools of resources to achieve a consolidated view of them throughout your IT environment.

Finally, as long-envisioned, the network of systems itself becomes a computer, the platform for services to run on. The technology that enables the building and management of virtual operation in a network is known as grid computing, such as described in the IBM grid computing portal. The grid computing solutions are based on standards defined by the Open Grid Services Architecture (OGSA), standards that help you create computing grids that include heterogeneous sets of hardware virtualized infrastructure components that are used for computing, storing, and accessing of large amounts of data.

In addition to these fundamentals of virtualization, appropriate management services are required to govern the whole system. We won't describe all of the available solutions, but you can refer to relevant literature such as the IBM Redbooks series on grid computing. We do want to point out, however, that these services are gaining importance in the context of SOA.

3.2.2. Service-Level Automation and Orchestration

Just as important are the technologies that facilitate automated management of required service levels and policies in any system running services. Imagine a world in which computers could monitor, analyze, and fix their own problems without much or any human intervention. Making a parallel with the human body, the autonomic nervous system is the part of the nervous system that controls body functions that are not under conscious and direct voluntary control. Computer self-management and self-healing require autonomic computing capabilities across your entire IT infrastructure. Similar to any manufacturing processes, self-managing computing systems can control and orchestrate an increasingly complex and expensive IT environment using appropriate autonomic management systems. This also evolves the role of IT professionals to include expanded responsibility that is no longer focused on basic errors or problems that are handled automatically, but rather to focus on higher levels of operation management such as the overall quality of services delivered.

The specific services that fall under the service-level automation (SLA) and orchestration subdomain are the automated services for problem management and system failure recovery, workload balancing and resource management, and system security services and data provisioning. The provisioning in this context also covers the installation and deployment of services in the system. Products like IBM Tivoli Intelligent Orchestrator provide these services.

All these services contribute to make IT systems resilient, responsive, efficient, and secure. Autonomic computing technologies today are evolving rapidly and enable construction of SOA-based solutions that reflect high service levels.

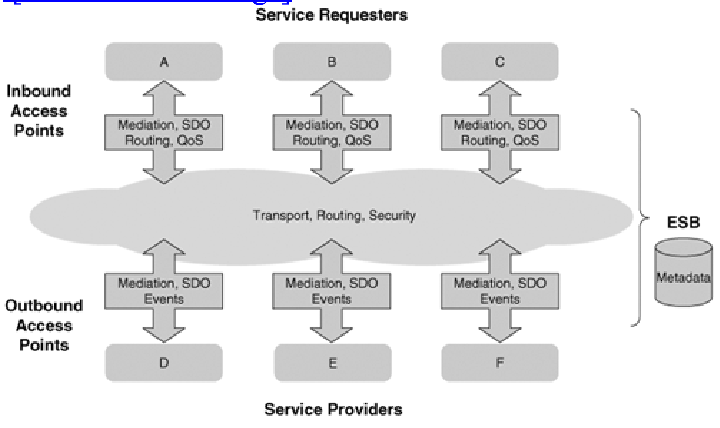
3.2.3. Utility Business Services

3.3. The Enterprise Service Bus (ESB)

To realize the scenario of an automated, self-managed SOA, the ESB is an essential architectural element. As such, it is a core part of the ODOE reference architecture, which is introduced in Section 3.5. Figure 3.2 is a conceptual view of the ESB, representing some service requesters at the top and service providers at the bottom for schematic purposes. In the real world, distributed environment requestors and providers can be located anywhere.

Figure 3.2. The ESB concept.

[View full size image]



An ESB is a core intermediary, a means to tie services together into componentized, logical sets. These sets reflect the structure of the business and are designed for distributed, widespread use across the enterprise. The logical grouping and design of each service component ensures that there is minimal heterogeneity in the business semantics exposed by the services. Each service forms a facade for the components or other technologies that implement the business logic. The essential infrastructure services that an ESB provides are transport, quality-of-service-based routing, mediation, and gateway services.

The ESB is actually an architectural construct that can be designed and deployed in a manner that will parallel the business processes environment. The bus can be implemented in various ways, such as with classical messaging, EAI, and brokering technologies or by using platform-specific components such as the service integration buses in J2EE systems (such as WebSphere Application Server). The ESB can also be a combination of both EAI and application server technologies, but the implementation should not affect the overall architecture. The selection between possible implementations will be the result of an initial architecture assessment, including existing IT infrastructure, skills, and processes in the evaluation.

The ESB acts as the intelligent, distributed, transactional, and messaging layer for connecting applications, diverse data, and other services that are commonly distributed throughout an enterprise computing infrastructure. It attenuates its core synchronous and asynchronous messaging backbone with intelligent transformation and routing capabilities, and it ensures that messages are passed reliably. The ESB enables developers to invoke and use business functions in components, regardless of API or protocol, by using them as services defined by a standard interface description based on the Web Services Description Language (WSDL).

WSDL separates the abstract descriptions of service interfaces, the reusable protocol bindings for the service, and the actual deployed endpoints offering the service. It is inherently extensible and offers extensibility elements for ports and bindings, allowing several different protocols to bind to the same service definition, if needed.

Literally, any unit software task can be described in Web Services Description Language (WSDL) directly, regardless of the protocol that is used to expose the service API (for example, Microsoft.Net, remote Enterprise Java Beans [EJB], Common Object Request Broker Architecture [CORBA] based components, applications listening on a Java Messaging Services [JMS] described queue, local entities such as Java bean classes, batch programs, executable files, and so on).

Effectively, the ESB is organically assembled from the different application and data components that provide

3.4. SOA Enterprise Software Models

To support all of the characteristics we just described, you need a software model supported by an architectural framework that provides key architecture principles and promotes the integration of new and existing services and business goals.

In this section, we focus on the various models that exist in the SOA space. This includes semantic models that address the business content and dynamics of the interactions, programming models that create or choreograph services, and off-the-shelf integration packages that expose services. We then explore the specific project and method aspects in Chapter 4, "SOA Project Planning Aspects," and Chapter 5, "Aspects of Analysis and Design."

3.4.1. Industry Models

Many industries' communities (some listed in this section) have already initiated their journeys toward SOA and offer modeling or implementation accelerators in the form of information models, services models, or business process models. These initiatives cover one or more SOA aspects such as the data formats, Web services interfaces, business process models, or even business services components with the associated contracts.

The following are some of the major initiatives that have made this level of progress toward an SOA:

- ebXML: Specifications and standards from the Organization for the Advancement of Structured Information Standards (OASIS) target the creation of an electronic marketplace in which enterprises can meet and interoperate in their businesses. In a recent move, ebXML has created the Electronic Business SOA Technical Committee.
- OAGIS: In the automotive industry, the Open Applications Group Integration Specification (OAGIS) also integrates SOA aspects.
- NGOSS: Because of its inherently heterogeneous and multiparty environment, the telecommunication industry has gone a long way in its services definition. The TeleManagement forum standards body has defined contracts in its New Generation of Operation Support Services (NGOSS) specification that are the business services together with business process models and decomposition and the associated information and data model.
- IFX: In the banking industry, the IFX Forum addressing public financial services is adding Web services to its existing services and information model.
- IFW: Also in the banking industry but targeting all financial domain aspects, IBM has defined a complete Information Framework (IFW) that includes business process and models.

3.4.2. Platform-Independent Realization

The SOA software model implementation must first focus on neutrality by using platform-independent standards. Because such standards address heterogeneous environments, platform-independent modeling, meta-data, and operational standards are necessary. They allow the interchange of software artifacts and facilitate interoperability between tools, applications, middleware, and data stores across platforms.

The Internet is composed of heterogeneous technologies that successfully interoperate through shared protocols. One of the key attributes of Internet standards is that they focus on protocols rather than on specific implementations. The need to implement to a common shared protocol standard prevents individual vendors from imposing their own proprietary standard across the entire Internet. Open-source software development of Internet protocols, for example, plays a crucial role in preserving the interoperability of software vendor

3.5. The IBM On Demand Operating Environment

The various domains of an IT operating environment have different concerns and requirements. When planning an SOA, the constraints from each domain will lead to different service component requirements. To address this need for an enterprise-wide, service-oriented IT system, IBM defined the on demand operating environment (ODOE) as its reference architecture that supports industry-wide best practices in SOA. As such, it takes a top-down approach to the development of a secure process- and service-oriented enterprise architecture that can be extended beyond a single enterprise into interactions with those of other organizations.

It is important to point out that starting with this high-level view does not mean that you need to rework your entire IT infrastructure. This is an architectural framework that enables you to build and maintain a platform-independent service model that maintains feedback from the many domains and that can be automated, monitored, and audited.

The ODOE encourages compatibility by enabling you to encapsulate the functions of existing operational systems and create a federation of heterogeneous software components. In this case, an existing operational system is anything that does not already have a service-based interface and interacts using well-known service protocols. This environment enables you to compose operational applications from atomic or coarse-grained business services to operate as part of the SOA. The SOA allows these coarse or atomic services to be accessed through various channels such as a visual user interface, through a portal on the Internet, or even through programmatic access using protocols such as Web services.

Web services, as defined by the industry standards and the profiles defined by the WS-I organization, are the basis for enabling these SOAs. Users can easily mix and match functions from heterogeneous environments into a single application built on Web services. You can have a portal that features Web services-based public interfaces that connect to yet another Web services portal to provide the application functionality. These backend services can be aggregated to support a broad spectrum of requirements, from aggregated component construction into groups of services, to transactional management of these service interactions.

Enabling functional access to Web services either through portals or service interfaces is not enough to provide a production-ready SOA. Such a system must also integrate operational aspects such as ensuring and monitoring service levels while using appropriate security protection.

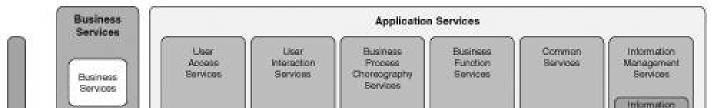
The ODOE provides protection of the internal infrastructure and sensitive data from inadvertent or fraudulent accesses. In some cases, it might also be necessary to protect even the existence of a service from unauthorized access.

The on-demand computing model applies at various levels in the IT stack. At the system level, the components are system objects (computing capacity storage and files). At the application level, the components are dynamically integrated application modules that constitute sophisticated, yet much more flexible, applications. At the business level, the components are business objects, defined for particular vertical industries or, more generally, to apply horizontally across industries. With a basis on industry standards, the on-demand computing model is appropriate not just for intra-enterprise scenarios but also across an industry ecosystem. The use of open standards makes it possible and applicable to the heterogeneous environment to create just such a cross-industry business computing ecosystem. It makes it possible to conceive and implement complete, end-to-end, business-process integration.

As depicted in Figure 3.7, the ODOE architecture defines four top-level domains: Business Services, Application Services, Enterprise Service Bus, and Infrastructure Services. We provide a basic overview of these domains and services, but you can find more details in a separate ODOE architectural overview paper published on IBM developerWorks.

Figure 3.7. The IBM on-demand operating environment.

[View full size image]


Business Services
Business Services
Application Services
User Access Services
User Interaction Services
Business Process Choreography Services
Business Function Services
Common Services
Information Management Services
Information

3.6. Summary

This chapter first refined the aspect of SOA characteristics such as loose coupling. We then introduced the four main domains of an SOA model. Using the example of the ODOE, we identified each of these domains and layers and their service characteristics. We then focused on the ESB as a pivotal element of the model. We also looked at the programming models that are necessary for the domains that involve programming. Finally, we looked at the domain of information management, which is an essential component. This domain can usually be supported by off-the-shelf software packages or specialized middleware.

Implementing successful SOA will need a selection of methods, models, and a careful analysis of the requirements and its value. This can be done only with structured roadmaps and control of the SOA implementation, which we discuss in the next chapter.

3.7. Links to developerWorks

A.3.1 Brown, K. and Ellis, M. Best Practices in Web Services Versioning. IBM developerWorks, January 2004. http://www-128.ibm.com/developerworks/webservices/library/ws-version/.

A.3.2 Beatty, J., et al. Service Data Objects. IBM developerWorks, November 2003. http://www-128.ibm.com/developerworks/java/library/j-sdo/index.html.

A.3.3 Selvage, M., Wolfson, D., and Handy-Bosma, J. Information management in Service-Oriented Architecture, Part 1: Discover the role of information management in SOA. IBM developerWorks, March 2005. http://www-128.ibm.com/developerworks/webservices/library/ws-soa-ims/.

A.3.4 IBM alphaWorks. Ontology-Based Web Services for Business Integration. September 2004. http://www.alphaworks.ibm.com/tech/owsbi.

A.3.5 Schmidt, M.-T. and Kalyana, S. S. The On Demand operating environment—Architectural Overview. IBM developerWorks, August 2004. http://www-106.ibm.com/developerworks/ibm/library/i-odoe1.

3.8. References

Bhaskaran, K. and Schmidt, M. T. WebSphere Business Integration: An architectural overview. IBM Systems Journal, Vol. 43, 2-2004. http://www.research.ibm.com/journal/sj43-2.html.

Dijkstra, Edsger W. A Discipline of Programming (facsimile). Prentice-Hall, 1997.

Dijkstra, E. W. Go to Statement Considered Harmful (reprint). ACM Classics of the Month, October 1995, reprint from Communications of the ACM, Vol. 11, No. 3, March 1968, pp. 147–148.

Ferreira, L. and Berstis, V. Fundamentals of Grid Computing. IBM Redpiece (REDP-3613-00), 2002. http://www.redbooks.ibm.com/abstracts/redp3613.html?Open.

Ferreira, L., et al. Introduction to Grid Computing with Globus. (SG24-6895-01), IBM RedBooks, 2003. http://www.redbooks.ibm.com/abstracts/sg246895.html?Open.

Globus.org. Open Grid Services Architecture. http://www.globus.org/ogsa/.

Gottschalk, K., et al. Introduction to Web services architecture. IBM Systems Journal, Vol. 41, 2-2002. http://www.research.ibm.com/journal/sj41-2.html.

Graham, S., et al. Building Web Services with Java—Making Sense of XML, SOAP, WSDL, and UDDI, 2nd Edition. Sams, June 2004.

IBM. Autonomic Computing Offerings. http://www-03.ibm.com/autonomic/index.shtml.

IBM. Virtualization Engine. http://www-1.ibm.com/servers/eserver/about/virtualization/.

Jacob, B., et al. Enabling Applications for Grid Computing with Globus. (SG24-6936-00), IBM RedBooks, 2003. http://www.redbooks.ibm.com/abstracts/sg246936.html?Open.

OASIS. Web Services Architecture. http://www.w3.org/TR/ws-arch/.

OASIS. Web Services Resource Framework. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf.

OASIS. Web Services Security. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss.

W3C. Web Services. http://www.w3.org/2002/ws/.

Chapter 4. SOA Project Planning Aspects

"Adventure is just bad planning."

—Roald Amundsen

The architectural consideration of SOA in the preceding chapter offers advice on what directions to choose and how to define the strategic goals for an SOA project. This chapter takes the next step toward execution by focusing on how to plan an SOA project. The topics in this chapter constitute the best practices we have uncovered for forming a project office (see Section 4.1), how to define the phases of SOA adoption, the need for and mechanisms of SOA governance, and finally, the various project roles and how they interact with each other.

This is not intended to be a complete template for a project plan, nor do we intend to show the optimal organizational structure for the parties involved in SOA projects. Based on our vigorous experience with different clients in various industries around the world, we are fully aware that there is no one-size-fits-all solution, nor is there a perfect approach to building an SOA for any scenario. An organization's specific circumstances will dictate its individual needs for project structure and plans. This chapter simply proposes ideas that you can adapt based on your scenario.

The first step is to establish the project office.

4.1. Organizing Your SOA Project Office

As seen in the preceding chapters, SOA implies a greater focus on business value. Business models are described as modular processes. This is achieved by breaking down the business and respective IT systems into components, providing reusability and modularity. Componentization, in this context, applies not just to software systems, but also to the business units across the enterprise and the organization of the enterprise in question. Implementing an SOA project involves not just a consideration of how the project is implemented in an IT infrastructure setting, but in the end, it also results in a business transformation process across the whole enterprise.

To accomplish your project, you first need a roadmap to guide the strategy for your SOA adoption. To build your SOA adoption roadmap, you need to identify who is involved in the SOA project. These individuals should come from the contributing cross-business-unit teams. The actual teams you involve will depend on the level of SOA adoption you choose (see Section 4.2).

Depending on business value analysis and the consequent prioritization of business objectives and services, the team defines "what to do," "how to do it," "who should do it," and "how success is measured." The SOA project team creates the rules, processes, metrics, and organizational structures needed for effective planning, decision-making, steering, and controlling the SOA endeavors. They define the common business service model, the common core processes and business components involved in the SOA project, and the core set of assets that they will use.

Building a suitable team for an SOA project requires a careful avoidance of making radical changes to existing team strategies because such changes can unduly disrupt the culture in the workplace. However, at the same time, the teams need to align with the SOA goals, which usually cut across business units. In addition, the SOA project might need to adopt a management structure—especially at larger IT shops with substantial project goals—to manage the development processes for implementing components or to expose existing applications or legacy functions in terms of the appropriate service granularity.

Achieving the right organizational structure is one of the critical challenges in implementing SOA. At organizations new to SOA, one often encounters strong resistance to change that keeps the focus on short-term successes rather than directing appropriate business transformation to align with the business challenges.

Mature SOA organizations, on the other hand, span business lines and the boundaries of roles while achieving interdisciplinary coordination. However, starting small can help to mitigate risk by allowing you to choose a well-scoped and focused services-integration project that has a modest plan for organizational evolution. A cross-unit, organization structure can address all the aspects of the SOA. Based on our experience, this structure should include the following:

- SOA business transformation architecture council: This team is in charge of gathering the business requirements, performing business domain analysis and process engineering analysis, and identifying the necessary business components, services, and process modules. Instead of following a strict top-down approach, the council should use a mixed approach in blending top-down, bottom-up, and goal-based methods to ensure appropriate services identification. In particular, this team ensures that the exposed granularity of the defined services matches the business requirements and specifications—matching business components to IT components as services. More details on granularity issues and associated services layers are described in Chapter 5, "Aspects of Analysis and Design."

- SOA technical architecture board: This team ensures the alignment of business and IT, following industry and enterprise standards, and technically ensures that exposed services match the requirements for evolution and reusability as defined in the general guidelines for the enterprise IT development. Its members are well versed in emerging industry trends, state-of-the-art technologies, and standardization efforts. They are responsible for framing the technical enterprise architecture blueprints (the master IT plan for the enterprise), identifying niche architecture patterns, and promoting reusability principles. They work closely with the SOA transformation team.

-

4.2. SOA Adoption Roadmap

An SOA strategy should not be a big-bang replacement of an existing IT environment; rather, it should be a progressive and evolutionary roadmap. Often an overall replacement is impossible when the majority of people in the IT organization are busy maintaining the running systems. Therefore, the roadmap should reflect an iterative process.

An enterprise has several options for entry points into a service-oriented architecture. These options identify how much the SOA model penetrates into the business and defines levels of adoption. The options are as follows:

- Initial adoption: Enterprises that want to reduce risks initially go through a technology validation and a readiness assessment that analyze the technical and business impact in a defined scope. Eventually, the business and technical value realized from this scope can be extrapolated to actual implications for the organization; this usually translates into a deeper commitment to move to SOA. It involves early pilot tests consisting of creating and exposing services from business operations contained in new or existing applications. These tests are used for an early validation of several decision points such as the following:
    - The capability to transform existing legacy systems. This might include technical solutions such as messaging, adapters, and connectors, or it might lead to partnership with vendors that can provide products for a service-oriented integration.
    - The non-functional requirements capabilities such as performance, security, manageability, and the availability of tooling.
    - The organizational structure required to support an evolution of the enterprise, especially one that addresses skills gaps and institutes governance structures.
- Line-of-business adoption: At this level, the enterprise will identify a line of business and prioritize processes where the agility and flexibility that SOA offers will increase business value. Of course, the enterprise might have already defined these priorities or have a critical business issue to resolve. In these cases, you still need to assess the SOA applicability to solve the important issue. This involves a broader initial assessment phase and the identification of key metrics and critical success factors.
- Enterprise adoption: This level of adoption involves the construction of a business view of a service-oriented enterprise, with a complete prioritization of projects based on business value followed by the architecture and implementation phases. You need to categorize enterprise activities into separate business domains and components that constitute the enterprise. This categorization might already exist within an enterprise or an industry model (for example, the telecommunication eTom model from the TeleManagement forum) that has already-established categories. At this stage, you should establish an SOA governance council with the required empowerment to monitor, define, and authorize changes to services within the enterprise.
- Enterprise-and-partner-network adoption: At this level, there is a broad transformation of existing business models or the deployment of new business models involving not only the enterprise, but also its business partners, suppliers, or customers. The enterprise can then select the roles that are appropriate for delivering its value, becoming a service provider, consumer, broker, aggregator, matchmaker, or any combination of those roles.

For each of the prioritized business services and components, the roadmap follows the typical phases of IT project development, with inception, elaboration, implementation, and test and production phases, as typified in the Rational Unified Process™. However, each of these phases includes new activities that relate to the service

4.3. The Need for SOA Governance

Enterprises using SOA can adapt to target broader connectivity and increased revenues; on the other hand, doing so requires restructuring applications for greater flexibility and lower costs. This requires the alignment of the business and IT value chain, as described in Chapter 2, "Explaining the Business Value of SOA." With this evolution, the enterprise will also need to adapt the way the business and IT units interlock and define a new way of reflecting business requirements in terms of IT applications. For this reason, organizational governance plays a more prominent role than before. The following sections provide guidance on establishing key governance functions for operating an SOA.

4.3.1. SOA Governance Motivation and Objectives

The business operations and the underlying IT infrastructure in an organization must react very quickly to rapidly respond to new business opportunities. Business units have to prioritize new IT services that have to be designed and managed as part of highly integrated and complex enterprise architecture. To achieve this, we discuss in the following sections a set of key governance functions for a successful SOA roadmap.

Governance provides an overarching structure to prioritize and then support the enterprise business objectives on a strategic, functional, and operational level. The governance model defines "what to do," "how to do it," "who should do it," and "how it should be measured." It defines the rules, processes, metrics, and organizational constructs needed for effective planning, decision-making, steering, and control of the SOA engagement to meet the enterprise business needs and challenging targets. As previously indicated, the SOA project team is responsible for creating this governance model.

The following are key questions that can help define the appropriate governance structure:

- What business change does the enterprise expect from SOA? Is it a better use of its existing infrastructure at lower costs, does it target new business and interaction models, or does it target both?
- Which roles, responsibilities, structures, and procedures exist to allow business prioritization and IT funding, planning, steering, and decision making?
- How can you develop skills and leadership competency?
- Which principles and guidelines are necessary to optimize the alignment of business and IT?
- What is the appropriate way to structure the business-to-IT relationship while keeping consistency and flexibility to allow the organization to quickly adapt to new changes?
- What is the appropriate level of standardization of services, the service definition, and the description?
- How do you control and measure services and service providers? What key business performance indicators do you need to monitor? Who should monitor, define, and authorize changes to existing services?
- How do you decide on a sourcing strategy for services?

We believe that an accepted and formalized governance model is crucial to successfully achieve business objectives, so we will define important governance functions in the following sections. For fast and high-level acceptance, it is essential to start from the existing enterprise structure and adapt it to the SOA roadmap.

4.4. SOA Technical Governance

With SOA, you can expect that business process cycles will be different from vendor product cycles. As a result, it is inevitable that, in the case of long-running or long-lived processes, you will need to support scenarios in which different versions of a business process exist concurrently on a changing infrastructure. Managing this challenge has implications throughout the project development lifecycle, not just for the runtime but also for the tools and methods used to define business processes within an enterprise.

You can manage the challenge of the dichotomy between business process cycles and product cycle by doing the following:

- Reducing the impact of changes by modularization
- Achieving middleware independence by defining the explicit process state
- Monitoring and handling business exceptions

Each of these topics is discussed in the following sections.

4.4.1. Reducing Impact by Modularization

Just as services can have different levels of granularity and permutations in the enterprise, processes also can have such granularity. This granularity appears when processes are designed as a composition of individual process modules. Each module offers a service interface and manages its own particular state internally. It then becomes much easier to change parts of the processes by developing new process modules that are selected from existing services using policies.

4.4.2. Achieving Middleware Independence with Explicit Process State

Current business process middleware engines maintain their process state internally. This dependency ties the process instances to the particular middleware engine, sometimes even to a particular version of the middleware. To avoid this, business process designers should elevate the explicit state beyond the engine level at each process step that leads to a waiting state until an external event arrives.

Thus, there is a need to be able to maintain and communicate state as distributed across the SOA. One particular programming model support for capturing these state descriptions is the set of specifications included in the WS-Resource Framework (as published on IBM developerWorks). These specifications allow the programmer to declare and implement the association between a Web service (a process module) and one or more identified, datatyped state components called WS-Resources.

A.4.2

4.4.3. Business Exceptions Monitoring and Handling

Even if the enterprise has spent a significant amount of time and effort to understand and model its business processes, undoubtedly unplanned business exceptions can still occur. A fully automated, services-oriented infrastructure that is capable of supporting any such exceptions to the business processes is unrealistic. This means that all business processes and their supporting infrastructure should be designed to allow manual recovery and control. Furthermore, for each business or technical domain, the organization should identify individuals that can handle such exceptions and act on the infrastructure. In most process and services identification modeling activities, the focus is on delivering mainstream models and a few variations. The

4.5. SOA Project Roles

The notion of handling exceptions in a manual fashion brings to light the key significance of people involved in SOA projects and the roles they might play. SOA projects involve many familiar project roles: project manager, business analyst, architect, developer, security specialist, and system and database administrator. However, these roles were created for different purposes and might have different inherent meanings based on the organization's viewpoint. To appropriately structure an SOA project, you need to consider that these roles might need to evolve to match the componentization and decomposition of the applications into services. In addition, implementing an SOA might also call for some additional roles.

In the following sections, we explain the functions of roles, and then we discuss roles versus skills in a project, followed by the phases in the project where these roles might apply. We then describe how these roles can be extended under SOA, as well as the new roles that follow. Finally, we examine some of the interactions among the various roles in SOA projects.

4.5.1. The Function of Roles

There is no single, global, standard definition of all IT jobs, not to mention SOA projects. Many jobs require certifications, a defined knowledge base, or cognitive tests; however, these certifications cannot always truly prove a candidate's applied knowledge, skill transfer, and creativity aspects. Because the team size, the workload, the types of work, and the subjects needed to solve IT problems vary greatly across companies, industries, and geographies, flexible teams with some specialist members are necessary. To coordinate such teams, the leaders (the architects and project managers) have to demonstrate the aforementioned capabilities beyond mere subject knowledge.

In this book, we use the notion of roles because doing so brings some order to the chaos. Roles are related to project phases and define an abstraction layer separate from actual job descriptions and assigned human resources. All project team members take on one or more roles.

Roles are a common concept in project management and design methodologies. The role concept establishes a commonly understood vocabulary, which has proven to be a powerful instrument at project-initiation time. A role does not imply that a specific individual person must execute the tasks that are associated with the role; instead, it implies an identity (an individual, a team, an organization, and so on) that fulfills that part of the process.

4.5.2. Roles and Skills

Identifying the roles involved in your project will certainly help, but finding the right people with appropriate skills is crucial. Services typically are technologically lightweight and simple; this is part of their power. With this technology, islands of heterogeneous systems can be more easily opened up for collaboration. However, along with these new possibilities come new sources of errors. SOA projects may be a new kind of project for your organization, but chances are they will not be a simpler kind of project.

An SOA project team setup should reflect these specific issues with the inclusion of appropriate skill levels and talents. We highly recommend a mix of practitioners who have experience with different platforms, different technical problems, and different skill domains. This is especially important for the SOA architect. If such a person is not available to the team, it would be feasible to have additional (part-time) co-architects to fill the gaps.

4.5.3. Project Phases

Development projects have different phases, requiring different skills and collaborations throughout the lifecycle, and SOA projects are no different. Independent of your organization's choice of the individual methodology used (waterfall approach or others), most projects will generally include the following development phases:

- Requirements engineering

4.6. Summary

After the SOA governance model has been launched, it sets the stage for the services modeling and architecting phase, which also includes the design of the necessary supporting IT infrastructure. This step requires the use of a formalized method. The next chapter covers the best practices and the modeling and architecture method we have captured from many of our SOA projects.

4.7. Links to developerWorks

A.4.1 Balzer, Y. Improve your SOA project plans. IBM developerWorks, July 2004. http://www-106.ibm.com/developerworks/webservices/library/ws-improvesoa/.

A.4.2 IBM. The WS-Resource Framework. IBM developerWorks, March 2005. http://www-128.ibm.com/developerworks/webservices/library/ws-resource/ws-wsrfpaper.html.

A.4.3 Zimmermann, O. and Mueller, F. Web services project roles. IBM developerWorks, June 2004. http://www-106.ibm.com/developerworks/webservices/library/ws-roles/.

4.8. References

Belbin, R. Meredith Management Teams, 2nd Edition. Elsevier Books, 2004.

Belbin, R. Meredith Managing without Power. Butterworth-Heinemann, 2001.

Bieberstein, N. Application Development as an Engineering Discipline: Revolution or Evolution? IBM Systems Journal, Vol. 36, 1-1997.

Hess, H. M. Aligning technology and business: Applying patterns for legacy transformation. IBM Systems Journal, Vol. 44, 1-2005. http://www.research.ibm.com/journal/sj44-1.html.

IT Governance Institute. IT Governance definition. http://www.itgi.org/template_ITGI.cfm?Section=Purpose&Template=/ContentManagement/HTMLDisplay.cfm&ContentID=14697.

MacMillan, K. and Downing, S. Governance and Performance Goodwill Hunting in Strategy Dynamics, Henley Management College, 2001.

Meredith, Jack R. and Mantel Jr.,Samuel J. Project Management—A Managerial Approach, 3rd Edition. John Wiley & Sons, 1995.

Nadhan, E. D. Service Oriented Architecture Implementation Challenges. EDS Solutions Consulting, position paper, April 2003. http://www.eds.com/services/innovation/downloads/so_architecture.pdf.

Telemanagement Forum. Homepage. http://www.tmforum.org.

Chapter 5. Aspects of Analysis and Design

"Every bright thing has been thought of before. One must just try to think of it again."

—Johann Wolfgang von Goethe

SOA has been developed and implemented in many ways by enterprises during the last decade using custom approaches for analysis and design. The increasing focus within the IT industry on widespread deployment of SOA is bringing greater attention to the practicalities of introducing specific service-oriented analysis and design methodologies and technologies.

In this chapter, we will explore some of the analysis and design considerations that apply to the development of service-oriented systems—those based on open-standard concepts and technologies as well as industry-tempered methods and well-tried software engineering principles.

Chapter 4, "SOA Project Planning Aspects," discussed the considerations for establishing SOA projects, and the roles needed as services are developed and deployed. In Chapter 6, "Enterprise Solution Assets," and Chapter 7, "Determining Non-Functional Requirements," we will continue to explore the SOA compass headings, with discussion focused on reusable architectural patterns, non-functional requirements, and qualities of service.

5.1. Service-Oriented Analysis and Design

The IT industry has been innovating techniques and methodologies for developing reliable, efficient enterprise software for the last four decades. Several models for construction of such software have evolved. Simple procedures, structured programming, data-flow programming, message-oriented programming, and object-oriented programming are just a few of those that are still in widespread use today.

Although these paradigms introduced new concepts and principles, each was built on the approaches that preceded it. For example, modularity is one of the engineering qualities associated with structured programming, and information hiding (encapsulation) is one of the qualities associated with object-based programming. In keeping with this evolution in thinking about software construction, there has been an evolution in techniques and tools for analysis and design and an evolution in runtime execution technologies.

The good news is that, throughout this lengthy period of evolution, a number of sound principles, acceptable constraints, and best practices have been learned within the IT industry and embodied in modern software. This evolving framework for architectural design carries forward as new models emerge.

Service-oriented, event-driven programming is the latest in a long succession of ideas to emerge. It incorporates many of the advantages of earlier models, builds on valuable lessons learned in previous eras, and introduces valuable new concepts, patterns, and practices.

A.5.1

The question is, what must we consider in order to analyze domain requirements and identify the best candidates for implementation as software services, and what must we consider when designing services that fulfill those requirements in the best way possible? The answer is that many well-established and familiar considerations still apply, but interestingly, SOA introduces some new ones that we will focus on in this chapter.

5.1.1. On Modeling

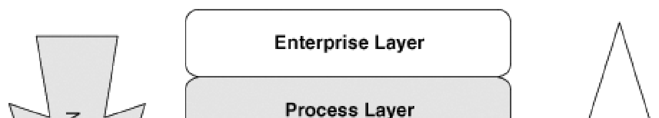
It is especially interesting to note that, independent of the industry focus on SOA, there has been an increase in focus on modeling. The OMG has published industry standards for model-driven architecture (MDA) and the Unified Modeling Language (UML) for use in development methodologies for commercial and other types of software. Others have proposed standards and technologies for business process modeling (BPM).

What used to be called "analysis and design" in previous technological eras is now often called "modeling." This development suggests a new level of formality and rigor in theoretical foundations and recognizes that there might be a continuum of definition, refinement, and transformation activities for analyzing requirements, developing architecture and design, and generating software code for target execution platforms. Some would argue with the suggestion of rigor, but there is no doubt that modeling is here to stay as a powerful technique for developing SOA solutions.

5.1.2. Layers of Abstraction

Much has been written about the value of abstraction and layering as techniques for thinking about software in an enterprise. They help organize thoughts, discussion, and documentation and provide focus at the most appropriate level of detail (see Figure 5.1).

Figure 5.1. SOA layers of abstraction.



5.2. Service-Oriented Analysis and Design—Activities

Analysis and design is based on good decision-making according to the best principles, acceptable constraints, and best practices. Service modeling based on good abstraction into layers, information hiding, loose-coupling, and strong cohesion can yield good services. However, just as modeling has changed the focus of analysis and design, so have activities come to the fore in any discussion about development methodology. Activities are the building blocks for the methods used when constructing software, whether they are sequential (waterfall), iterative (spiral), or extreme (agile).

A number of activities have been identified as essential to any method used for architecting and building an SOA. The activities are specific to services and do not replace any of the more traditional activities for developing objects and other application components. This section discusses some aspects of the most-often-used, service-specific activities.

5.2.1. Identifying Services

Perhaps the most obvious activity is identifying services. This can be done using input analysis, top-down domain decomposition, bottom-up existing systems synthesis, and meet-in-the middle techniques. Input analysis involves reviewing business models and existing system documentation for the business process in focus. The following are questions that arise frequently in this phase:

- Are the business drivers and goals for the SOA project articulated and quantifiable in terms of key business performance indicators?
- Have the business processes to be realized been named and described at a level of detail that is sufficient for architectural decision-making at the IT level?
- Have the existing and future non-functional requirements been documented with any unresolved pain points?

If the input from business modeling and the existing system documentation is not sufficient, it must be possible to schedule additional analysis activities as there is no point in proceeding with the modeling without a solid baseline.

5.2.1.1. Top-Down Analysis

Top-down business analysis techniques, such as IBM Component Business Modeling (CBM), may be the best starting point for identifying services. Using such techniques, it is possible to map major business functionality against industry templates in order to identify a "heat map" of core business processes that are candidates for SOA transformation or reengineering. Once the candidate business processes have been identified, they may be modeled to capture requirements.

Existing object-oriented analysis and design techniques can then be applied to identify and define services required. However, a higher viewpoint needs to be taken in most situations because a process-wide object-oriented analysis might lead to a large, unmanageable object model. SOA does not dictate a particular decomposition style, and alternative methods have described the use of different styles. For example, the IBM service-oriented modeling and architecture (SOMA) method starts with a functional decomposition step for this activity.

A.5.4

5.3. Summary

Techniques for analysis and design have evolved dramatically in the last four decades. Recent developments have focused on modeling and the use of metadata as the basis for methods and tools that cater to a wide range of activities, from business process modeling to automated generation of executable software. In this chapter, we have discussed service analysis and design (modeling) with emphasis on the most important considerations: encapsulation of service functionality, loose-coupling for service providers and consumers, and strong cohesion for SOA longevity. In the next chapter, we will discuss appropriate ways to capture decision-making patterns as templates for use across enterprise SOA projects.

5.4. Links to developerWorks

A.5.1 Zimmermann, O., Krogdahl, P., and Gee, C. Elements of Service-Oriented Analysis and Design. IBM developerWorks, June 2004. http://www.ibm.com/developerworks/webservices/library/ws-soad1/.

A.5.2 Beck, K., Joseph, J., and Goldszmidt, G. BPM: Learn Business Process Modeling for the Analyst. IBM developerWorks, February 2005. http://www-128.ibm.com/developerworks/webservices/library/ws-bpm4analyst
.

A.5.3 IBM. Patterns for e-Business. IBM developerWorks, 2004. http://www.ibm.com/developerworks/patterns/
.

A.5.4 Arsanjani, A. Service-Oriented Modeling and Architecture—How to Identify, Specify, and Realize Services for Your SOA. IBM developerWorks, November 2004.
http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/.

A.5.5 IBM, Microsoft, BEA, and SAP, WSPF: Web Services Policy Framework. Standard specification proposal, IBM developerWorks, July 2003.
http://www.ibm.com/developerworks/webservices/library/ws-polfram/summary.html.

5.5. References

Bennett, K., et al. Service-Based Software: The Future for Flexible Software. Paper submitted at Asia-Pacific Software Engineering Conference, 5-8 December 2000, Singapore. http://www.service-oriented.com/publications/APSEC2000.pdf.

Bhattacharya, K., et al. A model-driven approach to industrializing discovery processes in pharmaceutical research. IBM Systems Journal, Vol. 44, 1-2005. http://www.research.ibm.com/journal/sj44-1.html.

Booch, Grady. Object-Oriented Analysis and Design with Applications, 2nd Edition. Addison-Wesley Professional, 1993.

Fowler, Martin. UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd Edition. Addison-Wesley Professional, 2003.

Grossman, B. and Naumann, J. ACORD & XBRL US—XML Standards and the Insurance Value Chain. Acord.org, May 2004. http://www.acord.org/news/pdf/ACORD_XBRL.pdf.

IBM. IBM Service-Oriented Modeling and Architecture. IBM Business Consulting Services, white paper, 2004. http://www.ibm.com/services/us/bcs/pdf/g510-5060-ibm-service-oriented-modeling-arch.pdf.

IBM, IAA. Insurance Application Architecture, 2nd Revised Edition. 2004. http://www.ibm.com/industries/financialservices/doc/content/bin/fss_iaa_gim_06-29-04.pdf.

Jacobson, Ivar. Object-Oriented Software Engineering: A Use Case Driven Approach, 2nd Edition. Addison-Wesley Professional, 2005.

Jacobson, Ivar, Ericsson, Maria, and Jacobson, Agneta. The Object Advantage: Business Process Reengineering with Object Technology. Addison-Wesley Object Technology Series, 1994.

Jacobson, Ivar and Ng, Pan-Wei. Aspect-Oriented Software Development with Use Cases. Addison-Wesley Object Technology Series, 2004.

Kloppmann, M., et al. Business process choreography in WebSphere: Combining the power of BPEL and J2EE. IBM Systems Journal, Vol. 43, 2-2004. http://www.research.ibm.com/journal/sj43-2.html.

Koehler, J., et al. Declarative techniques for model-driven business process integration. IBM Systems Journal, Vol. 44, 1-2005. http://www.research.ibm.com/journal/sj44-1.html.

Latimore, D. and Robinson, R. Component Business Modeling: A Private Banking Example. IBM Business Consulting Services white paper, 2004. http://www.ibm.com/industries/financialservices/doc/content/news/newsletter/1061213103.html.

Mellor, Stephen J., et al. MDA Distilled. Addison-Wesley Object Technology Series, 2004.

OASIS, BPEL: Web Services Business Process Execution Language. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel.

OMG, MDA: Model Driven Architecture. MDA Guide 1.0.1, 2005. http://www.omg.org/mda/.

OMG, UML: Unified Modeling Language. Standard Specification. http://www.uml.org/.

Shlaer, S. and Mellor, S. J. Object Lifecycles—Modeling the World in States. Yourdon Press, 1992.

Stevens, W. Software Design—Concepts and Methods. Prentice-Hall, 1991.

W3C, RDF: Resource Definition Framework. http://www.w3c.org/RDF/.

Chapter 6. Enterprise Solution Assets

"Great things are not done by impulse, but by a series of small things brought together."

—Vincent van Gogh

Enterprise architects and consultants are faced with some serious challenges when formulating an IT solution. First of all, large-scale enterprise solutions are never easy to construct. They usually involve bringing congruency to a collection of heterogeneous legacy systems; breaking down silos of processing centers; migrating applications from niche, unviable products and homegrown platforms; and replacing ad-hoc enterprise plumbing that currently holds up a shaky infrastructure—this, by itself, is a Herculean task. Add to this the new demands for enhanced, non-functional requirements and qualities of service that CIOs traditionally expect when embracing emerging technologies, and you have opened a Pandora's Box that contains all the deferred wish lists of the past. You will have to accommodate the wish list requests in the proposed solution.

The lack of well-established, industry-tempered, reusable assets, such as templates, patterns, best practices, and road maps, further exacerbates the architect's dilemma. This is largely due to the pace at which enterprise technology progresses today. New specifications bloom with a rapid regularity (along with a slew of acronyms and buzzwords), and vendors rush to incorporate and provide support for them, churning out frequent product releases to keep pace. Hence, it is difficult to formalize, create, and maintain truly reusable assets because the underlying technologies and the product mappings keep shifting. Therefore, the assets become obsolete at the same pace with which technologies emerge. Additionally, there are organizational barriers in creating and maintaining assets because of the early investment to initiate an asset repository and because the return on investment is realized gradually. IT decision-makers are also concerned with whether asset creation and reuse would slow the usual IT development processes and the shelf life of any assets that are created.

This chapter describes how to identify and create architectural assets that are reusable across enterprise projects. These assets can be used for accelerating solution transition, articulating best practices, and capturing architectural decisions.

6.1. Architect's Perspective

Before describing enterprise solution assets (ESAs), it is important to understand the broad issues that an enterprise architect faces in any reasonably sized project. These issues range from selecting the appropriate architectural methodology to effectively mapping software products to the solution.

6.1.1. Selecting the Architectural Methodology

The first thing an architect attempts to identify is a coherent methodology and a set of techniques to apply to the project. Some of the choices available are model-driven architecture (MDA), component business modeling (CBM), and Rational Unified Process (RUP). Chapter 5, "Aspects of Analysis and Design," covered these architectural facets in detail, so they are not discussed in detail here.

6.1.2. Formalizing Architectural Decisions

After analyzing the solution requirements, an architect can then look further to derive architectural artifacts, which can be used to create high-level solution components. He can then place them in a solution context. During this process, several alternatives are placed on the architectural palette, such as technology choices, application patterns, and vendor choices. Based on these alternatives, the architect has to make certain choices to meet the functional and non-functional characteristics of the eventual solution. Some of these architectural decisions are often encountered and "rediscovered" across other intraenterprise projects. Being able to capture and formalize these architectural decisions would provide a key reusable asset for future projects. This is especially true for SOA projects because defining common, shared services is a critical success factor.

Currently, there are no formalized methods to capture this architectural decision framework. Design patterns (such as Gang-of-Four patterns) primarily provide guidance only for design decisions. They provide low-level, object-oriented patterns that are targeted and applicable for the solution design and construction phase. However, design patterns have been available for a while, and their structure is well understood by architects. The ESA concept will reuse the nomenclature and structure of design patterns (with certain modifications) to preserve syntactic and semantic concepts so that they can be easily understood; it will then extend these concepts to formalize architectural decisions.

6.1.3. Identifying Architectural Best Practices

Because SOA best practices are still maturing, an architect usually spends a lot of time searching for existing best practices. These can then be evaluated for applicability to the enterprise solution at hand. Compounding this is the fact that these best practices apply at various layers: low-level technology (such as J2EE) practices, service-orientation procedures, industry vertical standards, business domain-specific scenarios, and others.

The following are some of the formalized options for such best practices currently available:

- J2EE patterns and blueprints have been extensively used for adopting architectural practices that leverage a Java-based middleware platform to implement solutions.
- IBM patterns for e-business provide a group of reusable assets to create e-business applications. They provide platform-agnostic guidance to create effective enterprise solutions quickly. The patterns are classified into the following:
    - Business patterns that are used to create simple end-to-end scenarios involving interaction among users, business processes, and data. They are the fundamental building blocks of most e-business solutions.
    - Integration patterns connect Business patterns to provide advanced functionality and create more complex applications.
    -

6.2. Enterprise Solution Assets Explained

An enterprise solution asset (ESA) is defined to assist the architect in creating reusable assets to primarily address the issues of architectural decisions and vendor product gaps, as described in Section 6.1. An ESA describes common problems and difficulties that occur frequently in architecting and designing enterprise solutions, and it proposes solutions to address them. As these assets are created, they will be classified and housed within a catalog.

As described in Section 6.1.2, the layout of an ESA closely resembles that of design patterns only to exploit its structural familiarity. In general, an ESA has the following essential elements:

- An enterprise solution asset name is a handle to uniquely describe an enterprise problem and is used as an index into the ESA catalog. It also contains a one-line statement to describe the intent of the asset.
- The problem synopsis captures the essence of the problem and describes where to apply the asset.
- The context describes the problem in terms of a concrete example and lays out the situation in which the asset might be leveraged.
- Forces summarize the architectural decisions and design considerations that led to the solution presented within the asset.
- The solution is the core of the asset, and it describes a general-purpose solution to the problem that the asset addresses. The solution section can also include technical specifications such as diagrams describing the solution model (for example, UML class diagrams, sequence diagrams, component diagrams, and so on) and descriptions of the participating components in the solution.
- The consequences lay out the implications—the pros and cons—of using the asset. They also propose architectural alternatives.

The solution section of an ESA also has accompanying executables and the associated usage documentation that are compiled as an RSA-compliant package. The ESAs described later in this chapter are derived from IBM engagement experiences with real-world SOA projects. They have been generalized and abstracted from the engagement specificities. These primarily provide ESA samples to help create your own ESAs.

6.3. A Catalog of Enterprise Solution Assets

We recommend that from the onset of an SOA project, you create an ESA catalog to publish the assets so that they are visible across the enterprise. Incorporating a taxonomy to classify the growing list of ESAs would be beneficial and would help promote the reusability of the ESA across other enterprise projects. This book is not intended to be an extensive ESA reference; however, to explain the concept, we describe the following ESAs:

- Multitiered disconnected operation allows an enterprise application to operate in the event of failures such as network outages and unreliable data availability. This is described in Section 6.6.

- The request response template provides an infrastructure facility for client components to control how data is requested. This allows a client to request only a data subset returned by the service. This minimizes data flow during a service request and also helps isolate the client from service version changes and enhancements. This is described in Section 6.7.

The IBM service engagement teams are gathering potential assets in ongoing customer projects. As the list of assets grows, they will be classified into various taxonomies such as problem domains and industry segments. These assets will be part of the IBM SOA integration framework. However, this chapter's primary intent is to provide guidance to create your own ESA and ESA catalog.

6.4. How Does an ESA Solve Enterprise Problems?

An ESA provides some key facets in solving enterprise architecture problems. An architect equipped with an ESA catalog is able to use formalized architectural decisions and well-engineered solutions. The key ESA benefits and value proposition are as follows:

- Productivity: By providing well-documented assets contained in an ESA catalog, an architect is able to use these assets to address the core solution requirements, reduce the project delivery timeline, and arrive at robust and resilient solutions.
- Consistency and standardization: By generalizing the specific problem or gap scenario from a particular project scenario, the ESA allows applicability to other enterprise projects. It also enables architects to consistently apply a standardized asset to solve similar problems within and across various industries.
- Risk mitigation: By leveraging the ESA, the architect is able to provide robust solutions for product and technology gaps identified during a project engagement. This reduces the risk of ad-hoc architectural decisions and solution weaknesses.
- Maintainability: An ESA will be well maintained in its lifecycle with controlled changes and relevant updates. This facilitates the delivery of further ESA enhancements and potential defect resolution in a quality-controlled environment.
- Knowledge and intellectual capital sharing: As the ESA catalog grows and evolves, it serves as an essential artifact that enables effective awareness and guidance on key enterprise architectural issues.

6.5. Selecting an Enterprise Solution Asset

Finding the correct asset to solve your enterprise architecture problem is difficult, especially if the catalog of ESAs is new and unfamiliar. The following list provides guidelines and advice for identifying the relevant assets that might be applicable to your situation:

- Scan the ESA name and intent area to narrow down your choices and read the context to ensure applicability to your problem.
- Study the assets applicable to your industry segment based on the catalog classification that is available.
- Check the ESA catalog regularly because more assets are introduced as the catalog is formalized.

6.6. Using an Enterprise Solution Asset

After you have identified the ESA that you need, you can leverage it for your enterprise architecture by following these steps:

- Read the asset for an overview, paying attention to the forces, context, and the consequence sections to ensure that the asset is applicable for the situation.
- Step through the solution in detail and understand the core components used in the solution, paying particular attention to the potential solution variations.
- Peruse through the technical documentation and code that accompany the asset.

6.7. Multitiered Disconnected Operation

Multitier disconnected operation allows an enterprise application to operate in the event of unreliable data availability due to failures such as network outages.

6.7.1. Problem Synopsis

Enterprises use applications or suites of applications requiring data that spans multiple infrastructure tiers. Although increasingly reliable, the networks connecting these tiers are never 100% reliable. In addition, there are cases when, even if connected, the data might not be available to the tier that requires it.

A fundamental problem facing enterprises is the capability to allow such multitier applications to continue functioning during events such as network failures. This asset enables a multitier application to operate in an environment without reliable data availability. Solutions that require multitier operation can leverage this asset.

6.7.2. Context

Scenarios from customers in the retail industry exemplify the need for multitier disconnected operation. Figure 6.1 shows a sample multitier retail enterprise. There is a point of sale (POS) application that must run at the POS terminals in a store (terminal tier), on the store server (store tier), and at the retail headquarters (considered here as the enterprise tier).

Figure 6.1. Multitier context for the retail industry.

[View full size image]

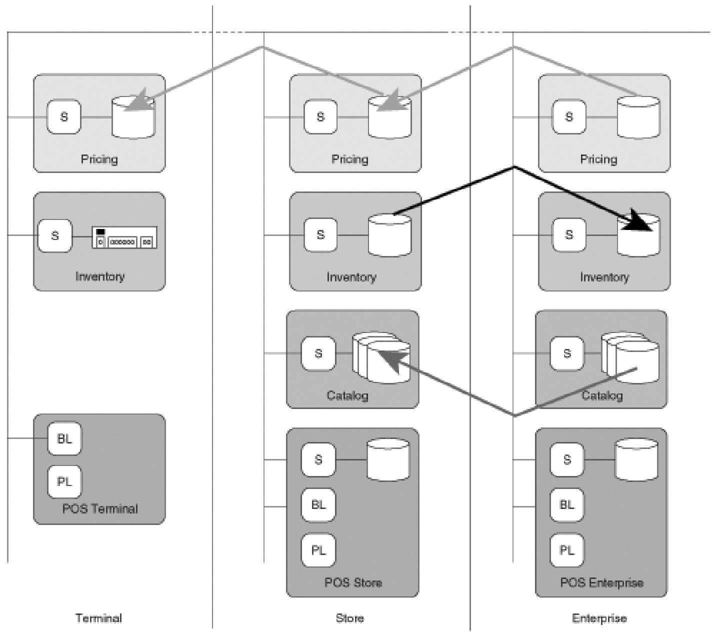


Figure 6.1 shows three databases with different characteristics. Pricing represents a read-only database (containing static lookup data) that must be available to all tiers. This is because a POS terminal must know the price of items sold, even if the terminal is disconnected from the store server. The store server acts as a POS application as well, in case all terminals are down. The light gray arrows in Figure 6.1 indicate that the pricing database is created at the enterprise and copied to the store server; the store server then cascades the data to the individual terminals.

The Catalog database represents another read-only database. It differs from the Pricing database because it does not need to be present at the terminals; the POS application can function without access to that data. The catalog is copied from the enterprise to the store level, which is indicated by the dark gray arrow in Figure 6.1.

The Inventory database represents a transactional database that must be available and updated at all tiers. The POS terminal needs to modify the inventory kept at the store server as items are sold or returned. Because it

6.8. Request Response Template

The request response template ESA is useful for decoupling the message schemas of a service requestor from their service provider.

6.8.1. Problem Synopsis

Web service interfaces are tightly coupled contracts with usage patterns mandated by the service provider. The asset decouples client usage of a Web service from the contract specified by a service provider.

6.8.2. Context

The following are some common problems due to the tightly coupled nature of service requesters and providers:

- Tightly prescribed interfaces implicitly determine the usage patterns for a service. Service providers build their own efficiency compromises into their interfaces, and clients are forced to accept them.
- Clients need more control over the data and service models for service responses.
- Clients want to circumvent large response message payloads associated with coarse-grain services.
- Service providers need to evolve the interfaces they provide without breaking existing clients.
- Clients would rather program toward abstract data and service models rather than directly to WSDL-specified interfaces.

Web service providers have a common problem when dealing with multiple clients: WSDL prescribes an exact interface for data types, messages, and operations that implicitly restrict clients to a usage pattern that is not necessarily appropriate. Service providers must choose the WSDL they expose, even though clients are the only ones who know the most efficient interface for their usage patterns. The result is an unattractive choice for a service provider: It must provide multiple interfaces for multiple clients (which are hard to develop and to maintain), or it must provide a single, one-size-fits-all, compromise interface. However, that compromise can be difficult to realize because service and data models might provide more or even less information than a client actually needs; the provider is left with another choice as to how to avoid message bloat (too much data) or overly chatty interactions (not enough data). Whether producing multiple interfaces or compromise models, the provider still might have imperfect information about the needs of its clients, forcing it to make an educated guess about the operations and data models it should expose.

6.8.3. Forces

These are the main forces in designing this asset:

- To build on existing Web service standards, enhance the existing specification, and do not reinvent the wheel.
- To use existing, well-known modeling tools that can define abstract interfaces (UML, for example).
- To increase flexibility, granularity, and maintainability of an application's Web service interfaces.
-

6.9. Summary

Enterprise solution assets are primarily intended to formalize architectural decisions and create reusable solution artifacts that can be leveraged across enterprise projects. It is recommended that project managers and architects accommodate the ESA tasks as part of project schedules. Managing and maintaining the ESA catalog should be placed under the SOA governance principles. As ESA catalogs mature with the addition of new ESA assets, it will also provide the capability to share ESAs across enterprises.

6.10. Links to developerWorks

A.6.1 IBM alphaWorks. Web Services Response Templates. http://www.alphaworks.ibm.com/tech/wsrt/.

6.11. References

Adams, J., et al. Patterns for e-business—A Strategy for Reuse. IBM Press, 2001.

Gamma, Erich, et al. Design Patterns, 1st Edition. Addison-Wesley Professional, 1995.

Kruchten, P. The Rational Unified Process: An Introduction, 3rd Edition. Addison-Wesley Professional, 2003.

OMG (Object Management Group). Model Driven Architecture. http://www.omg.org/mda/.

OMG (Object Management Group). Reusable Asset Specification. http://www.omg.org/cgi-bin/doc?ptc/2004-06-06.

OMG (Object Management Group), UML: Unified Modeling Language. http://www.uml.org/.

Sun Developer Network. J2EE Patterns. http://java.sun.com/blueprints/patterns/index.html.

Chapter 7. Determining Non-Functional Requirements

"There are two kinds of truths: those of reasoning and those of fact. The truths of reasoning are necessary and their opposite is impossible; the truths of fact are contingent and their opposite is possible."

—Gottfried Wilhelm Leibnitz

The non-functional requirements (NFRs) of a service-oriented business system address those aspects of the system that do not directly affect the functionality of the system but can still have a profound effect on how the business system is accepted by both service consumers and those responsible for supporting the system. The non-functional aspects of SOAs cover a broad range of topics, each of which can have a significant impact on the architecture. These topics can be classified into four major categories:

- Business constraints
- Technology constraints
- Runtime qualities
- Nonruntime qualities

In the context of SOAs, each of these categories has additional implications for target systems. The runtime qualities, for example, include those often specified in service-level agreements (SLA). This chapter examines each aspect in greater detail.

7.1. Business Constraints

Business constraints are non-functional requirements that can have a significant impact on the overall design or deployment of an SOA. These constraints include operating ranges, regulatory constraints, legal constraints, or standards established in specific industries.

7.1.1. Operating Ranges

Public services can be accessed from anywhere in the world, and this might imply that availability for such services is almost continuous (24 hours a day, 365 days a year). Enterprise services in an intranet that spans geographic regions may share this requirement. The enterprise service bus (ESB), an important component in any SOA, meets such requirements. The ESB can dynamically route service requests and responses to maximize availability of critical services based on applicable policies.

7.1.2. Legal Constraints

Legal and regulatory constraints are often forgotten when defining SOAs, but they can have significant impact on system design. For example, the law in Luxembourg requires that information about a banking customer be encrypted while it is in transit between service endpoints with no possible breaches by decryption. This implies that service consumer programs running on PCs of banks must encrypt information that cannot be exposed by intermediate application servers that would have to decrypt the information to process it. It also implies end-to-end certificate management through software installation on the bank PC machines.

Confidentiality and the need for encryption are not the only constraints of this kind. In certain countries, there are stringent laws regarding data privacy. SOA deployments in these countries must make provisions for an appropriate data-privacy-handling mechanism that includes solutions such as digital signing to ensure that data is not exposed. Another legal constraint that might arise in cross-border scenarios involves regulations and contracts that must apply when consumers and providers are located in different countries. For this reason, many public service providers install networking entry points in each of the countries they operate in, with provisioning cycles that include the contractual aspects that are specific to the country.

7.1.3. Industry Business Standards

Some industries have started to characterize the significance of SOAs within their domain. These industry-specific standards must be identified in advance of widespread SOA adoption because they influence many aspects of the architecture. For example, the Telecommunications Industry Telemanagement Forum (www.tmforum.org) has defined NGOSS contracts that are truly service models for operational support services. Some governments, such as New Zealand's, have already published standards on the Web for service delivery architectures that are for vendors who want to use their IT infrastructure. These provide a framework for delivery of a wide range of e-government services to citizens of that country.

7.2. Technology Constraints

Technology constraints are based on choices, decisions, and commitments to specific technologies in current and continued use in the enterprise infrastructure. These decisions and commitments to buy and use certain technologies can originate from business choices and relationships or from recommendations from IT technical teams. For example, choice of particular application packages, decisions to use OEM hardware platforms, and commitments to use industry-specific standards can impose constraints on future development of SOAs. The difference between these business-originated technology constraints and the business constraints defined earlier is that the former involves the use of specific products and technologies, whereas the latter involves the need for specific features in the implementation (which might be independent of the product or technology used to implement the features).

The business reasons for choosing technologies—and thus creating a technology constraint—can originate from the following:

- Business commitments to use a particular technology to support various internal departments, external suppliers, partners, or governmental organizations that dictate the need.
- Mergers and acquisitions that introduce additional technical infrastructures (from the acquired or other organization) that you might need to continue operating and supporting while attempting to integrate into your own organization's infrastructure.
- Budget limitations that restrict teams from acquiring or implementing new technology.

Technical teams within an organization might also make commitments to use a particular technology based on their evaluation and research. These technology-originated constraints might reflect technical choices that are based on the following:

- Adoption of existing or new technology standards (as opposed to business standards)
- A need to continue supporting an existing base of products (usually too tightly coupled to replace or upgrade quickly or easily)
- A need to continue supporting a technical design
- A lack of available technical knowledge or expertise to change the existing system or adopt a new one

Whether the technology choices are made for business or technical reasons, they can still result in the same types of technology constraints that impact the non-functional requirements of your architecture. These technology constraints include the following:

- Operating environment constraints that restrict changes to the infrastructure (hardware or software)
- Technical design/model constraints that restrict the technical teams to using or supporting an existing model or design and the data contained in those models and designs.
- Access constraints that restrict changes to interfaces and access mechanisms
-

7.3. Runtime Qualities

Runtime qualities explore all of the system aspects that are directly involved with system dynamics such as performance, scalability, transactional integrity, security, and fault tolerance. In SOA systems, runtime qualities are directly addressed by service-level agreements. SLAs are often based on end-to-end measurements—from consumer to provider involving all intermediary components. Consumers usually do not care how the internal systems perform within service provider boundaries and are more concerned with the performance of services in relation to the delivered quality at the endpoints.

7.3.1. Performance NFRs

Customers expect perceived service performance to be expressed as a combination of both response time and throughput. The choice of technical architecture components will have serious impact on the overall delivery capability within an SOA. For example, loose-coupling and interoperability often require the use of service messages based on the SOAP wire format on the HTTP protocol. A SOAP message can easily be many times larger than a traditional binary message. This results in greater bandwidth requirements. In addition, applications need to parse the content of service messages, converted from their portable, on-the-wire format into a format that can be more easily processed by endpoint programs. Some of the emerging Web services standards, such as Web Services Security, present further processing and size issues. Performance in SOAs is impacted by decisions made about the following:

- Service granularity and placement
- Binding choices
- Message parsing and data volume
- Security models
- Network bandwidth

These considerations are mainly focused on the protocol needed for service invocation, the amount of information that flows across a service interface, and the need for security-related networking interactions.

7.3.1.1. The Impact of Service Granularity and Placement on Performance

The granularity of exposed business services needs to be carefully designed. Services that are too fine grained lead to a greater number of interactions between endpoints; on the other hand, services that are too coarse grained can lead to unnecessarily large information exchanges. (Refer to Chapter 5, "Aspects of Analysis and Design," for the discussion about granularity.) Poor interface design can stem from designing interfaces that do too little with each message (sometimes called "chatty" interfaces). One reason for chatty interface design is that the message content model does not lend itself to complex, multipart messages. This is common in remote procedure call-based distribution models. The best practice to avoid these chatty interfaces is to have a rather coarse-grained interface with an information model that can dynamically adapt to the consumer's needs.

Mainframe systems often require the breakdown of service requests into sequences of lower-level service requests, or non-service-based interface requests into existing transaction systems (such as the previously mentioned CICS or IMS examples). You can achieve this breakdown using middleware components such as MQ Integrator Agent for CICS™, WebSphere MQ Integrator™ flows, or WebSphere Application Server™ using micro-flows. Mainframe transaction managers seek to minimize flows that are highly optimized for traditional styles of access, such as 3270-Terminal access over a systems network architecture (SNA) network. However, most services-oriented solutions require additional elements (such as connectors, adapters, encrypted data flows, or new data stream architectures) that are less optimized, and impose an apparent overhead on the execution of

7.4. Nonruntime Qualities

The non-functional requirements discussed so far have related to decisions made about the runtime components of an SOA. Nonruntime qualities explore all of the system aspects that are related to a system's lifecycle and that control aspects such as manageability, version management, and disaster recovery.

7.4.1. Manageability NFRs

As we progress toward delivering true SOAs, what becomes distinctly evident is the critical need for managing agents that operate deep within the operating environment, in conjunction with high-level controlling layers. A single management dashboard is often required to satisfy this need, and for this to work well, it must be part of a larger services management environment, not just a slightly modified environment for gateways and exchange management.

7.4.1.1. The Requirements for Services Management

To understand the overall effort and scope of requirements needed for services management, it is important to achieve a common baseline of IT functionality that the services management system needs to provide. This is not simply an issue of capturing SOAP fault alerts in a monitoring console, but rather a need to encompass a broad range of interoperable and loosely coupled services within a SOA framework. A proposed starter set of these functions should include the following:

- Security management: This includes the capability to manage individuals and roles for authentication ("Who are you?") as well as authorization ("Are you allowed to perform this function?"). As part of an overall policy management capability, authentication and authorization services need to be managed under broad security policies as they exist for an enterprise.
- Catalog support: The capability to catalog services with associated deployment metadata requires management of catalogs as components in a federated network of repositories. This includes authorization and authentication ("Can a requesting service 'see' all services available in a catalog?") so that entitlement to access is appropriately managed.
- Provisioning: This includes the capability to provision a service as well as provision additional capacity for a service (if a service is under duress). Capacity management should be included as part of the provisioning process.
- Configuration and versioning: Another essential function in services management is the capability to configure services as part of a policy. That configuration can be static or dynamic, depending on the terms and conditions of the contract for which the service is being exploited. Versioning is vital to ensure that the exploitation of services happens in accordance with prerequisite and co-requisite dependencies (such as between the calling service or program and the receiving service).
- Monitoring: The foundation of management, this includes the capability to manage service capacity thresholds, faults, errors, and otherwise predictable and unpredictable conditions in which valid processing did not occur.
- Performance and SLA monitoring: This is the capability to monitor service throughput metrics and capacity, as well as work with an intermediary such as a metering service to aggregate performance data and create input for SLA reporting.

Further details on the topic of services management systems are covered in Chapter 9, "Managing the SOA Environment."

7.5. Summary

In this chapter, we identified some of the non-functional requirements that SOA systems must satisfy. These requirements were categorized according to their source and significance as business constraints, technology constraints, runtime qualities, and nonruntime qualities. The intent was to illustrate some of the requirements found in typical SOA projects rather than to provide an exhaustive list.

Which of many non-functional requirements will dominate the architecture of your next SOA project is impossible to predict. Within a particular enterprise, it might be that industry standards and partner systems dominate the requirements. The most likely scenario, from our experience, is that existing systems and the technology choices made in previous years will dominate the non-functional requirements that must be accommodated.

7.6. Links to developerWorks

A.7.1 IBM alphaWorks. The Ontology-based Web Services for Business Integration. http://www.alphaworks.ibm.com/tech/owsbi.

7.7. References

Appleby, K., et al. Policy-based automated provisioning. IBM Systems Journal, Vol. 43, 1-2004. http://www.research.ibm.com/journal/sj43-1.html.

Buco, M. J., et al. Utility computing SLA management based upon business objectives. IBM Systems Journal, Vol. 43, 1-2004. http://www.research.ibm.com/journal/sj43-1.html.

Dan, A., et al. Web services on demand: WSLA-driven automated management. IBM Systems Journal, Vol. 43, 1-2004. http://www.research.ibm.com/journal/sj43-1.html.

Kreizman, G. and Fraga, E. E-Governmment Architecture: Development and Governance. (TG-14-6799), New Zealand State Services Commission, October 2001. http://www.e-government.govt.nz/docs/service%2Darch%2D200303/.

New Zealand Government Initiative. A Service Delivery Architecture, New Zealand State Services Commission. http://www.e-government.govt.nz/docs/service-arch-200303/chapter3.html.

Telemanagement Forum. Catalyst Spotlight: Model Driven Architecture for NGOSS. TeleManagement Forum, March 2005. http://www.tmforum.org/browse.asp?catID=1118&sNode=1118&Exp=Y&linkID=30310.

Chapter 8. Securing the SOA Environment

"Let every eye negotiate for itself and trust no agent."

—William Shakespeare

Security in a service-oriented architecture is a process of identifying areas of risk within an architectural model and providing trusted practices and countermeasures to mitigate those risks. As an integral component of an SOA solution, we need to understand the business-level concepts of risk and trust to explore what security services are required. Most enterprises already have security solutions that largely rely on established security controls, such as firewalls and virtual private networks (VPNs), to provide perimeter protection. As deployments of SOA solutions become more widespread, the process of securing the enterprise must become a fundamental part of an SOA development process. In addition, as an enterprise moves toward implementing SOA patterns, concrete implementations will require a move from the reliance on perimeter controls to a more granular view of security services and security architecture.

This chapter covers security in SOA in terms of understanding security concepts, the available standards and technologies in the industry, and how to structure your own SOA security model. It then examines SOA security in the context of a multi-organization environment, including federated security. Finally, this chapter presents an overview of relevant products for SOA security.

8.1. Architectural Considerations for an SOA Security Model

When evaluating existing deployments, you will need to understand any new risks and countermeasures that the SOA development model introduces. You should look at where security is implemented in an enterprise, either embedded in applications or as IT processes, and then evaluate how you can implement new security services to provide support to both application and enterprise SOA components.

From a security perspective, you will generally need to authorize any changes in software within an enterprise. This requirement can be met by a range of solutions, from clear business practices that identify the people in the organization that are responsible for managing software installation, to providing automated tracking of software patches applied to systems through an asset management process. The critical thing is to start capturing the business requirements for securing the change management process. These requirements fall into the following areas:

- Messaging characteristics—The capability to interact without depending on a particular messaging format, protocol, or interaction model allows a wider reach of integration capabilities.
- Transport protocol independence.
- Data format independence.

The requirements for transport and data independence usually imply that the organization needs a security model that can ensure the security of multiple, varied protocols. In an initial assessment for an SOA project, it is important to understand what security mechanisms currently exist and identify the enterprise-wide solutions versus the tactical solutions. The following are things to consider:

- Multiple interaction models such as synchronous and asynchronous models
- Message semantic independence, along with the role of tooling and runtime environments in providing semantic support
- Programming language independence
- Business model independence in the form of canonical format support, industry-based message exchange formats, and others
- Independence from business process standards such as RosettaNet, OAGIS, and so on

Each interaction model generally has its own mechanism to secure message exchanges. Depending on the number of standards that an enterprise must support, there might be a security requirement to support multiple mechanisms and to translate from one security domain to another. These mechanisms include the following:

- Support for various integration styles that provide architectural flexibility
- Unified user experience to provide a consolidated and singular interaction mechanism for users
- Application connectivity that addresses the communications layer that underlies all of the architecture

8.2. Concepts and Elements of Security

The basic elements of security include the concepts of integrity, confidentiality, identity and authentication, message authentication, session management, authorization, privacy, non-repudiation, and cryptography. These concepts directly impact the architectural and design plans for SOA security.

8.2.1. Integrity

Integrity of information refers to the state of a piece of information such that it is not altered in an unauthorized or unexpected manner. Maintaining the integrity of messages during an exchange between message partners allows all parties to have some level of assurance that the message has not been tampered with in transit. Integrity protection means that if a malicious user were to intercept and change the message content, it would be detected.

8.2.2. Confidentiality

Confidentiality of information refers to the state of such information with respect to any unauthorized or unexpected disclosure of that information. Integrity and confidentiality can apply to data or information at rest or to messages in transit. Confidentiality of messages allows partners to have some level of assurance that their messages have not been read by outside parties while in transit when it is important that sensitive details of a message not be revealed to outside parties.

Transient message confidentiality protection means that during the message transmission, its contents are undisclosed to malicious interceptors. Persistent message confidentiality mechanisms ensure that even after transmission of a message, the confidentiality of the message data (for example, credit card numbers, social security numbers, and so on) is maintained. If messages are logged, for instance, persistent confidentiality ensures that if someone were to access a log on a server, he or she would not be able to read the contents of the message.

Confidentiality and integrity also apply to information that is resident. For example, persistent data integrity applies to information such as persistent data or code, whereby signing techniques are used to ensure that the information is not altered. In general, you will need integrity and confidentiality techniques wherever you need to ensure that the state of information is not adulterated or introspected while in transit or in storage.

8.2.3. Identity and Authentication

Authentication is the process of validating a claimed identity. The authentication process evaluates a public piece of information (such as a username) and a private piece of information (such as a password, in theory known only to the user) to determine its validity compared to what the system knows. This private information is often referred to as an authentication credential. Authentication credentials might be based on something that is known (such as a password), something that is possessed (such as a private key maintained on a hardware token such as a portable USB storage device or fob), or an idempotent object that never changes (such as a fingerprint for a physical user).

After authenticated, systems and applications typically establish the privileges of the authenticated entity. These privileges, sometimes referred to as authorization credentials, are used to determine what this entity is allowed to do within an enterprise (also see Section 8.2.6). Thus, within the context of an authenticated secure session, the system can enforce authorized requests by the authenticated entity, and in some cases, it can also keep a record of everything the authenticated user attempts to do (for instance, an audit trail of his or her activities).

Consider, for example, an application used as part of an order-fulfillment process. This application might need to leverage other applications such as shipping schedules or updating stock levels. Access to shipping schedules might be restricted so that schedules for the next 24 hours can be viewed by the warehouse manager, but beyond that, schedules should be viewable only by order-entry clerks. Likewise, all updates to stock levels will be audited so that fraudulent activities can be monitored and stopped. In either case, having an authenticated identity, a set of authorization credentials, and an established trust relationship between the parties allows the appropriate actions to be permitted or denied, and it allows you to log requests (permit and deny) for audit

8.3. Implementation Requirements for SOA Security

The elements of security have different dynamics and impact on the architecture of an organization. Approaching security requirements from a service-oriented architectural perspective raises some new requirements on these elements. These requirements address how to coordinate security mechanisms across business partners and trust boundaries and are defined in terms of policies in the following aspects:

- When establishing trust between partners, the accurate definition of security policies covering transport, message, and data protection, including the security tokens in a request, will be essential (see Sections 8.3.1 through 8.3.6). In addition to having each policy, it is important to communicate these policies.
- After a business has its own policies defined, it will need to manage and coordinate any changes of security information (signing or encryption keys) across partners (see Sections 8.3.7 and 8.4).

8.3.1. Managing Security Policies

In advance of interacting with a business partner's SOA resources, you will need to determine appropriate security policies such as requirements for transport and message layer protection as well as data level protection. These policies can provide a component of an overall assurance strategy, leveraged by the individual businesses. They provide confidence that the SOA will allow only authorized, trusted business partners—with known, defined, legal liability relationships—to participate in cross-business transactions. In addition, you might need to implement such policies and monitor these implementations for business agreement or legal compliance.

8.3.2. Defining Transport Security Policies

Transport layer security refers to the type of protection offered in the actual delivery protocols involved in the interaction, such as using secure sockets layer (SSL) over Internet transactions. SSL is used to provide a confidential (encrypted) channel between two endpoints.[1] The result of this encryption is that the contents of any message flowing over this channel are not discernable to any observing, outside party. This encryption is based on keys stored within digital SSL certificates. These are long-term keys bound to an entity during initial SSL session establishment. Both parties use a common new encryption key that is possibly previously unknown for the purposes of encrypting all of the messages on the wire.

[1] One quick note: Because this information is encrypted and not readable, a form of integrity protection is implied. This follows from the fact that it is extremely difficult (practically as well as mathematically) to modify a message (violating its integrity) without being able to read it. So in a scenario in which it is not easy to decrypt/re-encrypt a message, integrity protection can be said to follow.

One side benefit is that SSL certificates, while defining a cryptographic key, also contain a binding of this key to a given entity. This enables software to use SSL certificates to also determine data origin authentication—if I am able to encrypt something for you to decrypt using the SSL certificate's key, and if I can determine that it is a valid certificate issued by someone I trust, then we both have established some level of trust.

After the SSL channel is terminated at the SSL endpoint, the messages are available in cleartext for anyone to see (and alter). In scenarios in which you need to terminate an SSL channel and forward the message to its intended destination service without using SSL (with intermediaries or gateways, for example), you can open a potential vulnerability into the system. This then leads us to a new requirement: message layer security.

8.3.3. Defining Message Layer Security Policies

Message layer security allows the system to protect the message body itself (transmitted in the HTTP message body, for example) in terms of integrity and confidentiality; this is on top of any protection applied at the transport level. Because this is applied to the message body, the protection is provided end to end. Furthermore, transport layer security might change depending on the network boundaries that the message traverses (different levels of security over the many individual networks constituting the Internet). End-to-end protection implies that the contents of the message cannot be read or modified at any point other than the required endpoint. In some

8.4. Standards and Mechanisms for SOA Security

This section introduces and describes the different security mechanisms available in the industry that can be used to address security requirements within a service-oriented architecture, both within a single organization and across multiple organizational boundaries. We will focus primarily on how this technology is implemented in the Web services (WS-*) security family of specifications.

The basis of security in Web services is in the WS-Security Roadmap published by IBM and Microsoft to describe the components necessary to address a secure Web services solution. This roadmap defines security elements as composable units that can be applied only when necessary to solve a particular problem. Thus, a security solution does not need the drag of additional functionality and components that are not required. We will describe how the parts of this roadmap fit together to help fulfill the security requirements previously described.

A.8.1

8.4.1. The Basic Security Standard: WS-Security

WS-Security refers to two distinct efforts: the initial WS-Security specification (published by IBM, Microsoft, and VeriSign) and the full WS-Security roadmap containing several defined specifications (including the WS-Security specification itself). This subsection focuses on the basic WS-Security specification.

A.8.2

The WS-Security specification was originally published in April 2002 by IBM, Microsoft, and VeriSign and subsequently was submitted to the OASIS Web Services Security Technical Committee (WSS-TC) created for it. The effort through the OASIS process led to an OASIS standard in March 2004 known as Web Services Security.

A.8.3

As described in the OASIS document, the specification provides three main mechanisms: sending security tokens as part of a message, message integrity, and message confidentiality. These mechanisms by themselves do not provide a complete security solution for Web services. Instead, this specification is a building block for other Web service extensions and higher-level, application-specific protocols to accommodate a wide variety of security models and technologies.

WS-Security enables you to apply XML security techniques (described in the following sections) to authenticate and secure message exchanges between a Web service requestor and a Web service provider. It uses signatures and encryption placed on a message and security tokens bound to the messages.

8.4.1.1. WS-Security Tokens

The WS-Security specification is agnostic to the type of token that is actually included within a message. WS-Security uses an XML approach to defining tokens that enables extensibility and supports multiple security tokens. This enables services to communicate in a secure manner and exchange security information across different implementations.

The Web Services Security: SOAP Message Security (WSS-SMS) specification defines three classes of token: username, binary, and XML. The username token is a simple representation of a username and optional password that represents the binding of commonly presented user authentication credentials into a simple XML-based token. The binary token format provides a means of representing binary-formatted information such as Kerberos tickets, X.509 certificates, and other non-XML-formatted (or -formattable) security tokens. Finally

8.5. Implementing Security in SOA Systems

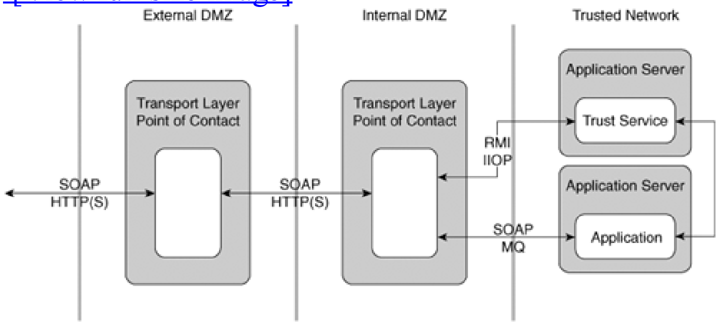
This section presents some common components based on the technologies described previously that you can leverage to provide security services in an SOA reference implementation. One of the key business transformations when implementing services-oriented architectures is to identify (common) security services and evolve business processes to take advantage of these services. Initial implementations will most likely consist of existing elements of the infrastructure enabled with basic security services in a hybrid environment.

8.5.1. Implementing Basic Security Services

The basic security architecture in Figure 8.1 contains two main security components, a point of contact (PoC) and a trust service, described in detail in the following sections. Figure 8.1 also specifically illustrates two different point-of-contact elements: a transport layer PoC and a Web services PoC.

Figure 8.1. Basic security architecture.

[View full size image]



A.8.7

In this hybrid example with multiple protocols and domains involved, SOA requests are bound to SOAP over HTTP requests. This approach enables the SOA architecture to reuse large parts of the pre-SOA architecture, including the HTTP (transport layer) components. This architecture shows an external demilitarized zone (DMZ)—a neutral zone between two security domains—containing a traditional HTTP (SSL) security PoC entry to the enterprise. When the software establishes a mutually authenticated SSL connection, it also includes a coarse-grained authentication of the request.

The architecture also has an internal DMZ, whereby a Web services PoC provides introspection of the message. This Web services PoC element can leverage trust services from within the trusted network as part of the actual message validation. In this example, we show that the Web services PoC uses the RMI/IIOP protocol to communicate with the trust service, but it also transforms the incoming SOAP/HTTP request into a SOAP/JMS request.

The trust service is the architectural component that establishes the trust relationship between partners. The trust is based on the validation of the incoming message, including the authentication of identities asserted in the message, and the mapping of identities and roles as established in existing business agreements with the partners. Each of these components is discussed in further detail in the following sections.

8.5.2. Implementing Point-of-Contact Services

One of the first tasks in defining a service-oriented architecture is to understand what the service provides and establish the interfaces for the service. A PoC service is responsible for the following:

- Authentication of requests and requestors

8.6. Non-Functional Requirements Related to Security

Implementing a security model can have a significant effect on how the SOA operates. In particular, it is important to consider the relation to two other non-functional requirements: performance and manageability.

8.6.1. The Performance Impact of Security

Implementing security does not necessarily force a performance trade-off. The following guidance focuses on minimizing the performance hit of security while not impairing the actual functionality. You can also use additional techniques, such as clustering, linear scalability of components, or leveraging hardware accelerators, to offset a performance impact. To this end, you need to consider several general guidelines:

- Focus on layered component architecture. This enables you to appropriately tune the environment to meet performance, scalability, and availability metrics. For example, layering security functionality away from the application's functionality enables you to tune optimal security performance independent of optimal application performance.
- Apply only a layered security approach. Security is expensive. Minimize this cost by relying on a layered security, focusing as much as possible on transport level security. For example, it is great to have end-to-end security, but a point-to-point architecture is more common. Rather than immediately incur the cost of end-to-end security, evaluate where these points are. If the endpoints are within an enterprise's control, use transport layer security to protect the communications while on the public network. If you have to have message-level security, be sensible about the cost of this security. For example, providing confidentiality at the element level is also expensive. If possible, rely on encrypting an entire message body rather than individual elements within the message body.
- Rely on a layered approach to authorization. In general, you should push as much authorization functionality as feasible close to the edge of the application domain, especially in a Web services environment in which message payloads are typically quite large. If you layer your transport layer authentication decision at the edge (at the external DMZ point of contact), you can prevent unauthorized business partners from flooding the enterprise with bad requests. If you place message layer authentication at the internal DMZ point of contact, you can ensure that only a business partner's authorized users are able to access the internal system; this again reduces the number requests the application will just reject. This authorization layering enables the application to best handle the security functionality in the application level, or data authorization at the data level to which the application has direct access.

8.6.2. Managing Security

After you implement a secure SOA solution, you will need to be able to manage this security infrastructure, including managing trust relationships, security tokens for authentication, security tokens for session management, and credential stores.

8.6.2.1. Trust Relationship Management

Trust relationships are usually derived from the use of cryptographic techniques such as public key infrastructure. Simply having (and exchanging) cryptographic elements across business partners is not sufficient to establish and maintain a trust relationship. Part of a trust relationship also involves asserting and accepting requestor identities and attributes across the established trust relationships. For this reason, it is integral to manage the tokens used to convey this information.

8.6.2.2. Security Tokens Used for Authentication

Services can use X.509 certificates as the basis for security tokens to convey authentication information to sign a message. A trust service's security token functionality validates security tokens to authenticate requestors.

8.6.2.3. Security Tokens Used for Session Management

8.7. Technology and Product Mappings

Given knowledge of technologies, implementation considerations, and related non-functional requirements, you should now take a look at different categories of technologies and types of products that implement the various PoCs, including those for the transport layer, Web services, trust services, and federation services.

8.7.1. Transport Layer Point of Contact

Transport layer PoCs, as separate from the Web services layer PoCs, tend to be HTTP-type components such as HTTP servers or HTTP proxy servers. These components have become familiar elements of most organizations, whether in intranet or internet scenarios.

8.7.2. Web Services Layer Point of Contact

The different security PoC implementations include XML firewalls and gateways, Web services gateways, and other security services implementations. Most early product vendors in the Web services space released XML firewalls or XML gateways. These products are typically proprietary implementations of PoC functionality focused on XML-based requests but not necessarily those based on SOAP. These products may accommodate multiple transport bindings, supporting HTTP-based, MQ-based, and other transport protocols. Web services gateways are a specialized form of XML gateway that focuses on Web services–focused protocols such as SOAP or RMI-IIOP as defined by WSDL bindings.

Security services are often included within an XML/Web services gateway. The initial implementations were typically proprietary and did not support the standards-based functionality described by WS-Security. With the standardization of the (SOAP-based) WS-Security specification by OASIS and the publication of the basic security profile by the WS-I, most gateway vendors now allow for interoperability of signatures and encryption on Web services messages.

8.7.3. Trust Services

Typically, trust services have not existed as standalone components or products. In general, they are bundled as part of security or federation services as they apply to a network of systems. A more advanced offering for both trust and federation is available in IBM Tivoli Federated Identity Management™.

8.7.4. Federation Services

To date, most federation service implementations focus on passive-client (HTTP browser) based approaches to single sign-on. This approach leverages publicly available specifications for single sign-on to allow users to seamlessly access resources within an enterprise and across trusted partners. Part of the exchange of information that occurs as part of the single sign-on protocol includes the exchange of security tokens, very similar to the security tokens that trust services handle.

The IBM Tivoli Federated Identity Manager is currently the only known product that includes a component- or service-based architecture, with both trust and protocol services. The trust service of this product features, including logical security token services, are designed to be pluggable and to provide the full range of functionality previously described.

8.7.4.1. Liberty Alliance

The Liberty Alliance Project was formed to deliver and support a federated network identity solution for the Internet that enables single sign-on for consumers and business users in an open, federated way. Liberty Identity Federation Framework (ID-FF) describes profiles for B2C-based single sign-on and additional functionality. Liberty ID-FF profiles include single sign-on (SSO), single log-out (SLO), register name identifier (RNI), federation termination notification (FTN), and identity provider introduction (IPI). Tivoli Federated Identity Manager implements a multiprotocol federation gateway with integrated management support for Liberty ID FF 1.1/1.2. The added federation capability enables enterprises to quickly and securely integrate identity-driven transactions with their middleware and portal platforms.

8.8. Summary

The basic principles of applying security in any software solution are about identifying the risks, evaluating them, and then formulating a plan to mitigate them. In SOA-based systems, these security principles are the same. However, additional factors are introduced because SOA proposes distributed services and decoupled application systems. To effectively secure these resources, a gamut of technology options and variations are available with accompanying performance and operational management overheads. Mapping the security risks to the solution options requires careful planning and should be conducted formally during inception of an SOA endeavor.

8.9. Links to developerWorks

A.8.1 IBM, Microsoft. Security in a Web Services World: A Proposed Architecture and Roadmap, IBM developerWorks, April 2002. http://www-128.ibm.com/developerworks/library/specification/ws-secmap/.

A.8.2 IBM, Microsoft, Verisign, Web Services Security. IBM developerWorks, April 2002. http://www.ibm.com/developerworks/webservices/library/ws-secure/.

A.8.3 Hondo, M., Melgar, D., and Nadalin, A. Web Services Security: Moving Up the Stack—Security in a Web Services World. IBM developerWorks, December 2001. http://www.ibm.com/developerworks/webservices/library/ws-secroad/index.html.

A.8.4 BEA Systems, Computer Assoc., IBM, Microsoft, RSA Security, Verisign, et al. Web Services Trust Language. IBM developerWorks, February 2005. http://www.ibm.com/developerworks/library/specification/ws-trust/.

A.8.5 IBM, BEA Systems, Microsoft, VeriSign, RSA Security, Web Services Federation Language. IBM developerWorks, July 2003. http://www.ibm.com/developerworks/library/specification/ws-fed/. IBM, BEA Systems, Microsoft, VeriSign, RSA Security, Web Services Federation: Active Requestor Profile. IBM developerWorks, July 2003. http://www.ibm.com/developerworks/webservices/library/ws-fedact/. IBM, BEA Systems, Microsoft, VeriSign, RSA Security, Web Services Federation: Passive Requestor Profile. IBM developerWorks, July 2003. http://www.ibm.com/developerworks/webservices/library/ws-fedpass/.

A.8.6 BEA, IBM, Microsoft, SAP AG, Sonic Software, Verisign, Web Services Policy Framework, IBM developerWorks, May 2003. http://www.ibm.com/developerworks/webservices/library/specification/ws-polfram/.

A.8.7 Bose, S. Using Web Services Security in WebSphere Application Server. IBM developerWorks, April 2004. http://www.ibm.com/developerworks/websphere/techjournal/0404_bose/0404_bose.html.

8.10. References

Basic Security Profile. Web Services Interoperability Organization. http://www.ws-i.org/deliverables/workinggroup.aspx?wg=basicsecurity/.

Benantar, M. The Internet public key infrastructure. IBM Systems Journal, Vol. 40, 3-2001. http://www.research.ibm.com/journal/sj40-3.html.

Hinton, H., et al. Federated Identity Management with IBM Tivoli Security Solution. IBM Redbooks (SG24-6394-00). http://www.redbooks.ibm.com/abstracts/SG246394.html?Open.

Liberty Alliance Project. Liberty Alliance Project Specifications. http://www.projectliberty.org/resources/specifications.php.

Makino, S., et al. Implementation and Performance of WS-Security. International Journal of Web Services Research, Vol, 1, No. 1, 2004.

Menezes, Alfred, et al. Handbook of Applied Cryptography. CRC Press, 1996.

OASIS, SAML. Security Assertion Markup Language. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security.

Schneier, Bruce. Applied Cryptography, 2nd Edition. John Wiley & Sons, 1996.

W3ORG, XML-ENC. XML Encryption Working Group. http://www.w3.org/Encryption/.

W3ORG, XML-SIG. XML Signature Working Group. http://www.w3.org/Signature/.

Chapter 9. Managing the SOA Environment

"The significant problems that exist in the world today cannot be solved by the level of thinking that created them."

—Albert Einstein

With the move to service-oriented architectures, the IT management landscape is changing. Existing solutions might no longer be adequate. Apart from managing the usual underlying physical and application resources, the radius now includes higher-level business applications and services. To match the increasing focus on providing business value, management solutions must evolve to measure this value.

Chapter 2, "Explaining the Business Value of SOA," describes why businesses should move to SOA and an on-demand enterprise model. The on-demand IT environment is characterized by flexible and highly responsive systems that have lower IT costs and higher utilization rates. To meet the challenges of business agility while lowering costs, best-practice business processes are being implemented to help boost productivity while increasing flexibility.

In the SOA approach, one such best practice requires additional management capabilities: service provisioning, securing services, monitoring the service status and health, understanding relationships among the services, and managing choreographed and aggregated services. This chapter describes these SOA management concepts and capabilities.

9.1. Distributed Service Management and Monitoring Concepts

Enterprise IT management and monitoring usually involves the use of well-known mechanisms and strategies in operation centers. These mechanisms can be loosely categorized into different operational models including triage, basic problem resolution, and resource-driven operations. Using these models, this section explains the evolution of the IT management models in use today and how they are changing with the transition to SOA. Each progressive step of this evolution (the management models) focuses on a higher abstraction of how a particular problem is viewed, moving from individual events to resource states (possibly aggregating events) to transaction workflows (aggregating resources), and finally to services (aggregating workflows) with service-level agreements.

9.1.1. Event-Driven Management

The first operational model generally performs event triage, focusing on receiving events and forwarding them on to the appropriate groups for resolution. These triage groups do little, if any, problem resolution themselves. They might review known problem logs to determine whether this event has a known solution and then implement the solution to resolve the problem. These groups are typically staffed by lower-skilled employees who might quickly rise into more senior levels after they have acquired enough expertise in a particular function.

The triage operational model can evolve into the next level of basic problem resolution within a short time period before transferring any events. Generally, these groups have an assigned time limit to either fix or transfer the event to the appropriate team. These groups are usually staffed at a higher skill level; more senior people might remain on this team to continue solving problems. These teams also provide input to the automation teams so that solutions to common problems can be automated as part of the environment. Variations on both of these models include a help desk to receive all initial phone calls and events. Additional business information can also be provided to the environment to prioritize by severity and business impact.

Regardless of the operational model, most IT operations centers are driven by events that trigger a call to action. These can be in the form of Tivoli Event Console™ events or new trouble tickets created through automation-integration functions. These systems can also prioritize the events, such as time sequenced, priority, business impact, or geography.

Some organizations have evolved their IT management operations to a resource-driven operational model. The workflows for these operations are driven by resource state changes. Resources can be real resources, virtualized resources, or clusters, and they typically are based on CICS, DB2®, UNIX®, Windows®, or capable network device systems. Operators select the next resource that needs attention and usually prioritize resources by time sequence or business impact. In these cases, the operations staff is no longer focused on dealing with raw events but on the states of resources. They have moved from strict problem resolution to some degree of anticipating or modeling problem scenarios and identifying solutions. Understanding the resource state and devising the operational tasks in managing the state requires higher staff skills.

9.1.2. Levels of SOA-Driven Management

The next level of IT management focuses on transaction-driven operations, in which transaction state changes drive the business workflows. Operations select the next transaction that needs attention, which is usually prioritized by time sequence or business impact. The operators no longer deal with resources or individual events; instead, they deal with the status of transaction flows. These flows are collections of events based on live or simulated transactions.

The next higher level of IT management focuses on business service–driven operations, in which whatever impacts business services drives the workflows. Operators select the next impacted service sorted by business priority or by service-level agreement(SLA) status. The key challenge in managing at this level is in understanding the business services and their components.

Many organizations are evolving through these various levels and models of IT management, moving up from simple event-driven management to more sophisticated models. This progression requires not only education and training but also organizational and philosophical changes within the organization. The transition to SOA drives these changes to focus more on business services and processes rather than isolated application views.

9.2. Key Services Management Concepts

Tools for managing services supplement traditional IT management products. The concept of looking at transactions versus resources across the IT infrastructure is a relatively new one and is better suited for a model that has loosely coupled components and applications. For example, some parts of the application might lie within the sphere of control for an enterprise, whereas other parts of the application are controlled by the IT organizations of its business partners or customers. To successfully deploy and support a services-based application, an enterprise needs a transactional paradigm as well as a resource paradigm to manage these interactions. These capabilities must enable enterprises to deploy and ensure the performance and availability of new service applications, even when parts of the overall application are not under its direct control.

Although introducing service orientation addresses many of the traditional problems of integrating disparate business processes and applications, deploying services-based applications introduces new complexities that you need to manage. These include the following:

- Monitoring the services layer for performance and availability
- Ensuring compliance with service-level agreements
- Managing security policies so that service-oriented applications can communicate securely, both internally and across organizational boundaries
- Tracking the dynamic interconnectivity of the loosely coupled components of the system to understand the performance, availability, and expense consequences
- Analyzing the root cause and correcting problems based on errors at the services layer
- Developing and testing applications composed of aggregated and already operational services
- Deploying, configuring, and updating a distributed, service-oriented application across organizational boundaries in a secure, reliable, and repeatable manner
- Tracking the business impact of the use of services on the business processes

The SOA-based infrastructure and its IT management tools must address these complexities. The following sections examine the impact of the use of middleware and changing standards on the IT management toolset.

9.2.1. Managing the Enterprise Service Bus

The enterprise service bus (ESB) described in Chapter 3, "Architecture Elements," is a term for the middleware component that facilitates an SOA infrastructure. It brings together the features of several integration paradigms into a single element of the infrastructure. The ESB is responsible for delivering messages across the network, routing them as necessary, with the qualities of service required by the end points.

To achieve this, an ESB can intercept and manipulate the messages as they flow through the bus, interposing logic that requires intelligent decisions about where to route them, what quality of service to apply, and what additional message processing might be required. Its capabilities include logging, pattern recognition, metering, transformation, message validation, customized routing, and policy-driven selection of endpoints. You can apply policies to the ESB that define which capabilities apply to different services, both in terms of the business and IT infrastructure requirements.

9.3. Operational Management Challenges

Services still have the traditional management challenges in areas of security, availability, configuration, and performance. Although an SOA can be developed with many different methods and technologies, the use of common open standards enables you to consider any of these services equally as yet another resource in the management domain. However, there are still some differences from existing management models that arise from service orientation.

One such difference between services and other managed resources is that the services are application layer components, whereas most system management tools are oriented toward middleware, network, OS, and hardware types of resources. SOA services, however, can participate and be reused in multiple different business process. Thus, the requirements for managing a service vary based on the business process in which the service participates. To service consumers, the only entity that is exposed and available for them to manage is the service (an endpoint) itself and not the actual processes and components that implement the service. The service-oriented application that implements the service will most likely be out of reach for the service consumer to manage, as this is the domain of the service provider.

Another difference is that in a non-SOA environment, processes and tasks might not be composed along well-defined function boundaries in their member applications. Therefore, it might not be easy to compose or decompose these applications. SOA-based services, however, with well-defined interfaces, align the business process steps closely with an entity (the service implementation) that can then be monitored and directly and discreetly managed. This difference implies that you might need a different view—likely an easier, more flexible one—and different tools for SOAs than the functionality available in traditional systems management tools.

To consider how these differences impact your enterprise, you might need to consider the use of different management perspectives and follow a phased approach to deployment.

9.3.1. Challenges with Respect to Management Perspectives

You can look at services management from either the service provider's point of view or that of the entire enterprise. There are common management challenges for both enterprise and service provider environments. There are also unique considerations for management within each environment. The SOA approach provides additional management challenges that cover the entire lifecycle of the application, from development and deployment to the manageability and maintenance of the services.

The common management challenges for both views are as follows:

- Monitoring and managing the availability of the services.
- Providing a service registry that contains information about the services that are deployed in the enterprise and their descriptions.
- Providing the information about the services to the architects. Architects can use this information to provide solutions to problems using existing services and by defining new services.
- Management of rules that help route service requests based on the content of messages.
- Maintenance of services from a virtualized location that provides a singular service view.
- Managing multiple versions of the same services and managing the service lifecycle.
-

9.4. Service-Level Agreement Considerations

Today, most IT departments offer their services to internal or external consumers. Although many such organizations today have negotiated SLAs in place with their external clients or OLA with their internal clients, these agreements tend to be technically focused (such as database size and growth in gigabytes, system availability, and others). More mature organizations have SLAs with their external clients that are more business focused (for example, response time for a transaction or number of transactions per day); however, these SLAs might be inadequate in terms of measurement and reporting, or they might utilize manual activities to carry out the SLAs.

Due to the lack of products supporting the measurement of business-oriented metrics, SLA metrics are frequently technology oriented and provide little useful indication of the service quality from the consumer's perspective. Different interpretations of the service quality by the provider and consumer of the service can cause dissatisfaction among both customers and service providers. Examples of useful business metrics for services would be assured delivery dates for supply-chain services or approval response times for financial services.

A key consistent measure for SLAs or OLAs, that both the service provider and enterprises can use, is the end-to-end transaction time. This is especially appropriate for SOA environments, which can share services and measure the end-to-end transaction time, both with synthetic transactions and with tools such as IBM Tivoli Monitoring for Transaction Performance (see Section 9.5). By measuring the end-to-end response time, you can include the involvement of all components and related services in the measurement. This provides a consistent measurement across the environment. The consumer does not care what the server availability is as long as the transactions are completed within the defined time period.

9.5. SOA Management Products

The challenges to IT operations management and the use of business-focused SLAs can be addressed with different types of tooling. Each of these tool types addresses the needs of a different area of service management, including business performance and business service management, IT application and resource management, transaction management, Web services management, resource monitoring, and middleware monitoring. To detail the functions of these tools by areas of management, we use the example of IBM software products that fit each area.

9.5.1. Business Performance and Business Service Management

Business performance management is the process of monitoring and managing the overall organization's business performance results over time. This leads to the popular notion of an executive "dashboard" that provides presentation and analysis of the overall business metrics of the organization in a real-time or pseudo-real-time fashion. Business service management is a similar notion that focuses on the level of all individual business services in the SOA.

The IBM Tivoli Business Systems Manager™ brings the entire enterprise together to provide a business view of the environment. IBM Tivoli Business Systems Manager software is intended to be a productivity tool for operations to enable an end-to-end business view of the infrastructure and application management monitoring. It uses the monitoring software that is currently installed (such as IBM Tivoli Monitoring™), and it maps alerts and events generated by the various monitoring software products to objects in its database, and these are then displayed on the product's console.

You can use IBM Tivoli Business Systems Manager to create business views. Usually the operations staff monitors various products and consoles to complete service obligations that might relate to missed critical messages. The IBM Tivoli Business Systems Manager console can reduce the number of separate individual consoles from multiple software products into a single consolidated console. Using this single user interface, the team can proactively monitor and manage problems, determine impacts, and aid in root-cause analysis. You can use IBM Tivoli Business Systems Manager to accomplish the following tasks:

- To deliver a higher quality of end-to-end services by identifying all impacted areas from a single action.
- To manage groups of related applications that form a business system.
- To manage groups of computing resources.
- To enable true critical path management.

This product represents both the physical resources and the business system representation. The physical resources are organized in a hierarchy defined by distributed placement rules or default table entries.

IBM Tivoli Business Systems Manager receives events from the mainframe environment using a component installed in each z/OS® (formerly S/390®) mainframe system and receives distributed events from the Tivoli Event Console or from other common listener feeds.

IBM Tivoli Service Level Advisor™ enables businesses to easily, proactively, and economically manage service levels across the entire organization for maximum uptime. This software's user-friendly, Web-based reporting capability and at-a-glance dashboard show the status of current service levels and predict future trends so that you can take preventive action now and avoid problems later. When a trend toward a violation is identified, IBM Tivoli Service Level Advisor software integrates with other Tivoli products and sends alerts to the IBM Tivoli Enterprise Console®, to the IBM Tivoli Business Systems Manager console, directly to your e-mail, or using Simple Network Management Protocol (SNMP).

With its automated service level agreement evaluation capability, IBM Tivoli Service Level Advisor software

9.6. Summary

This chapter presented an overview of the management concepts and key operational challenges in an SOA environment. Enterprise architects must understand the operational complexities in managing and monitoring distributed services and leverage the appropriate tools and their underlying infrastructure to address them. There are various technology components and emerging standards that paint the SOA management arena. Adopting an SOA-driven management to provide business service management propels an enterprise to a higher plateau of IT management and aligns it to be an on-demand business.

9.7. Links to developerWorks

A.9.1 IBM, Common Base Event Specification.
http://www.ibm.com/developerworks/library/specification/ws-cbe/.

A.9.2 IBM et al., Web Services Resource Framework, IBM, October 2004.
http://www.ibm.com/developerworks/library/specification/ws-resource/.

9.8. References

Dan, A., et al. Web services On Demand: WSLA-driven automated management. IBM Systems Journal, Vol. 43, No. 1. http://www.research.ibm.com/journal/sj/431/dan.html.

Farrell, J. A . and Kreger, H. Web services management approaches. IBM Systems Journal, Vol. 41, 2-2002. http://www.research.ibm.com/journal/sj/412/farrell.html.

IBM Tivoli. IBM Tivoli Products. http://www.ibm.com/tivoli/.

Kephart, J. and Chess, D. The Vision of Autonomic Computing. Computer Magazine, IEEE 2003. http://www.research.ibm.com/autonomic/research/papers/AC_Vision_Computer_Jan_2003.pdf.

Kreger, H., et al. Java and JMX: Building Manageable Systems, 1st Edition. Addison-Wesley Professional, 2002.

Leymann, F., Roller, D., and Schmidt, M. T. Web services and business process management. IBM Systems Journal, Vol. 41, 2-2002. http://www.research.ibm.com/journal/sj/412/leymann.html.

Naik, V. K ., Mohindra, A., and Bantz, D. F. An architecture for the coordination of system management services. IBM Systems Journal, Vol. 43, 1-2004. http://www.research.ibm.com/journal/sj/431/naik.html.

OASIS, WSDM. Web Services Distributed Management. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm.

Opengroup. The Open Group Application Response Measurement. http://www.opengroup.org/tech/management/arm/.

Chapter 10. Case Studies in SOA Deployment

"Experience is never limited, and it is never complete; it is an immense sensibility, a kind of huge spider-web of the finest silken threads suspended in the chamber of consciousness . . ."

—Henry James

Case studies help us learn from the lessons encountered during the implementation of other projects in the industry and help us recognize the concepts and elements of this technology independent of the particular implementation. The case studies in this chapter are in different vertical industries and implement different scales of SOA. The structure for each case study is fairly similar.

First we take a look at the scope of the project and the business case for undergoing the project. Next we consider the issues of development under the SOA paradigm, the goals and objectives of undertaking an SOA project, and the changes to roles, tasks, and the organization of the project. Then we address the concerns of business management as well as end users in terms of security, performance, and management of these services. We then outline the key technologies used in the implementation to achieve the SOA objectives. Finally, we examine the challenges and lessons learned from the case study.

10.1. Case Study: SOA in the Insurance Industry

The Standard Life Assurance Company is one of the world's leading mutual financial services companies. With more than 175 years of continuous services to its customer, Standard Life is the largest mutual assurance company in Europe, with more than 4 million customers and L100 billion in assets under management as of January 2005. In this case study, the Standard Life Assurance Company evolved its infrastructure from a focus on the reuse of code at the data layer to a mature implementation of a service-oriented architecture.

10.1.1. IT and Business Challenges

Standard Life needed to meet two main business challenges: enhance its business channels while developing new ones, and anticipate and adapt to rapid changes in the environment. In the course of business, Standard Life continues to face new challenges due to the increasing rate of change and complexity of operating its business. Furthermore, the challenge in 2005 lay in the need to deliver more from less, finding creative ways to manage or reduce its cost base without affecting its current services.

On top of these two business challenges, the technical implementation for Standard Life's architecture needed to satisfy a number of design requirements and considerations so that it could achieve the following technical advantages:

- Maintain design consistency
- Offer simplicity of design
- Provide supportability, scalability, and recoverability
- Identify common code patterns
- Provide clear guidance on technical models
- Support efficient problem diagnosis
- Advise business logic placement
- Allow impact-free change
- Increase system performance
- Enable an easy-to-develop environment
- Provide useful documentation

Standard Life's SOA architecture plan, and the supporting framework and infrastructure, began with the introduction of its Application Design Patterns defined between 1999 and 2001, at which time Web services standards and tools were not yet mature. Therefore, the company needed to build an implementation that could adapt the available technologies at hand.

10.2. Case Study: SOA in Government Services

Government systems prove to be some of the most complex types of organizations, involving many variations on organizational hierarchies (districts, departments, ministries, and so on), legislative or legal requirements, and a multitude of technical directions. This case study involves the Austrian Federal Ministry of Justice (MoJ) and its IT services provider, BundesRechenZentrum GmbH (BRZ), in an e-government project in 2002 to modernize and deliver effective government-to-business (G2B) and government-to-citizen (G2C) services. The Austrian MoJ has about 7,400 people, consisting of judges, prosecutors, clerks, and other administrators covering the district, appeals, supreme, and high courts. This does not include the 3,400 other persons who staff the nationwide system of 28 prisons.

10.2.1. IT and Business Challenges

The main challenge in this project was to connect multiple application service providers to a number of government databases to improve access to judicial records. In addition to the issues of handling legal document access and exchange between various agencies and partners, the existing infrastructure also had a specific proprietary set of APIs, which the new system had to maintain for backward compatibility.

The key goals for the project were to overlay a new, modern, flexible infrastructure over the existing database systems, improving the interfaces for any type of service provider (an xSP), and still maintain compatibility with users of the system with older tools—all to better serve the needs of the legal community.

The MoJ created a project stakeholders model identifying the different groups involved in creating this project. The overall steering committee worked with the project leaders from each team: MoJ personnel, BRZ personnel, and personnel from IBM. These project teams developed multiple working groups that interacted with experts from each key user group they intended to support.

The MoJ provided funding and subject-matter experts. BRZ provided development personnel with knowledge of the legacy application, as well as additional subject-matter experts. IBM provided development personnel, expertise in object technology, project management staff, and the company's experiences from previous international court system projects.

10.2.2. Technical Implementation

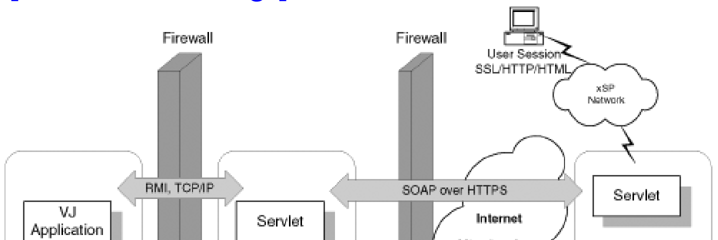
The previous customer model of the project, which the MoJ intended to continue, provided access to the court documents through the xSPs, whereby the xSPs were responsible for providing end-user access.

Customers of the system (that is, attorneys and other government-related businesses) pay a certain transaction fee to the xSPs, a portion of which is recovered by the MoJ. The fees vary by transaction, and the data access is logged at the xSP, with the customer getting charged according to the number of bytes transmitted.

The change to the technical infrastructure included a new SOAP-based Web services application server placed in a DMZ between the MoJ intranet and the xSP's systems, accessed through the Internet (see Figure 10.6). The Web services application server then connects to the MoJs backend systems consisting of IBM S/390® mainframes. The Web services application server handles SOAP requests from the xSPs described in XML and, as an intermediary, translates these to Java Remote Method Invocations over TCP/IP to internal applications on the MoJ intranet. The interfaces to these services that the xSPs access are described in WSDL.

Figure 10.6. The Austrian Ministry of Justice Web services project.

[View full size image]



10.3. Summary

These two case studies provided views into SOA projects that are independent of the protocol mechanism; that is, one project used Web services while the other built its own system. Both examples showed how intranet and extranet applications can exist in the same architecture. The lessons learned from each project included the emphasis that SOA needs a combination of technology processes and people to make it work and that it can deliver real savings with surprisingly smooth performance.

Chapter 11. Navigating Forward

"Following the light of the sun, we left the Old World."

—Christopher Columbus

In this book, we navigated through a broad range of issues that you can expect when building an SOA solution. As stated at the beginning, to go into detail for all possible aspects that you need to consider when transforming your business and IT systems is too large a task for a single book. Nevertheless, we think we have provided necessary and sufficient details on service-oriented architectures and why and how they are important for any business to operate flexibly in a global economy.

11.1. What We Learned

The foremost lesson to take from our experience in SOA projects is that you will need to deal with the whole enterprise—not just the IT departments, but anyone in the company involved and working as a service provider, service requestor, or both. This approach differs from those in the 1980s and 1990s, when the enterprise data model proved to be an enormous effort, getting people lost in details and minutiae. With service orientation, the focus is on providing an infrastructure that is as flexible as needed by the agile enterprises of today.

An SOA results in flexibility and not just within your own organization. It paves the way to build service-oriented business relationships with partners, suppliers, and customers. Another result is a saturation of sophisticated applications throughout your enterprise, building a competitive advantage that can shift the whole organization toward quicker reaction to customer needs, anywhere in the world. All this leads to the architectural principles described in Chapter 3, "Architecture Elements."

As we ourselves discovered more about service orientation, we came to understand that there are success factors that stem just from having a common understanding across the enterprise—you will need this understanding and a common language to facilitate the change toward greater agility. Though you can take incremental steps toward service orientation in many areas of your software, what you really need is a business transformation process under careful orchestration and guidance throughout the enterprise. On this point, we described (in Chapters 3 and 4) the aspects related to governance (IT and business), the need for special SOA project management, and the overall architectural blueprint that acts as an outline for the enterprise. These factors combined are what provide the real roadmap toward business agility.

Applying technology correctly can pave the way for the flexibility to change and continuous business innovation. This requires an infrastructure that supports self-defined, loosely coupled interfaces. Additionally, it calls for the use of tools from emerging technologies that incorporate existing assets through automation, virtualization, and integration. These tools also need to support self-defining, declarative semantics, as well as strong analysis and compositional techniques based on software engineering techniques like aspect-oriented programming.

As outlined in Chapter 4, "SOA Project Planning Aspects," the concepts, the services, and their inter-communication all need to be standardized in order to achieve loose-coupling and flexible interoperability. In addition, there also needs to be a change in how the organization develops these SOA-based systems. We showed that new roles come into play in the SOA model. We also encountered a need for closer cooperation between IT and line-of-business representatives.

We learned that most existing IT architecture can be a choke point for business innovation, as monolithic systems and applications cannot be easily reused, and that each generation comes with its own monoliths and assumptions. Merger and acquisition activities and the requirement for new ways of doing business electronically have proportionally grown the need for integration. Ad-hoc integration solutions that were used to link dedicated systems together are often custom-made solutions that create connections that are difficult to change and maintain.

Hence, a new architecture is required that overcomes these deficiencies. We showed in this book how the concepts of SOA—though not a totally new approach in principle—can help create an infrastructure, a development environment, and an integrated approach between IT and business groups within an enterprise and beyond. These SOA concepts provide the necessary base for flexible IT supporting an agile business.

To a certain degree, a lack of standards limits the capability to deliver meaningful interoperability. However, our collective experience has shown that the applicability of Web services and related industry standards can allow organizations to cross the barriers of programming and operating systems built up over the past decades.

The final lesson we learned is that the strength of your architecture is key to its success. A strong plan saves you from doing a big-bang replacement of systems with potential high risks of failure. Instead, a strategy of small improvements can help justify cost. Factors that play an important role here are common standards for communication and description (as given by SOAP, WSDL, and other WS-standards), as well as an understanding of industry and cross-industry semantics and the taxonomy of services (refer to Chapter 5, "Aspects of Analysis and Design").

11.2. Guiding Principles

This book represents, as with any other printed document, only a snapshot of time based on the accumulated knowledge and insights from ongoing projects at enterprises, in development, and in research laboratories. As with any new discipline, service-oriented architecture and the supportive on-demand operating environment will continue to evolve as subjects of further development.

Summarizing, we can say that an SOA enables flexible connectivity of applications or resources by doing the following:

- Representing every application or resource as a service with a standardized interface
- Enabling them to exchange structured information (messages, documents, business objects)
- Mediating the message exchange through a service integration bus
- Providing on-ramps to the bus for existing application environments

This allows quicker combinations of new and existing applications to address changing business needs and improve operational effectiveness by managing the topology of the application network.

The principles of SOA might sound like a simple approach; they might indicate a lack of sophistication or a "boil-the-ocean" approach. But this does not mean it is simplistic. An SOA, rather, is a smart way of allowing gradual and continuous improvements based on easy-to-understand patterns and a set of commonly accepted standards.

Based on our analysis of the collective experience of many IBM teams in more than 70 client projects in this area during the last two years, we can name seven guiding principles to consider when entering the SOA adventure:

1. SOA requires CEO- and CIO-level commitment (see Chapters 1, 2, and 4). SOA is not just a product for which standard IT ROI equations apply, nor is it just a new IT technology to apply. But to justify the transition toward SOA, consideration of both IT and business benefits are required.
2. The business team and IT team work hand-in-hand (see Chapters 4 and 5). SOA is all about flexible business processes that IT offers the means to implement. Adequate forms of business process modeling and industry or enterprise decomposition are the first critical steps leading to a well-defined set of services.
3. Avoid the "big-bang" approach (see Chapters 3 and 4). This means to start small by selecting a well-defined application or business process area. Then use the SOA blueprint to establish an initial target architecture, and finally, leverage existing data and backend processes using adaptor technologies.
4. Fully embrace the use of standards(see Chapters 3 and 6 through 9). Here we especially refer to open standards (for example, the Web services standards) and open source, providing a new and proven approach for a collective endeavor toward a greater target to master global business requirements in a quickly shrinking world.
5. Governance is critical for success (see Chapter 4). We recommend, as a first step toward establishing or enhancing the end-to-end implementation process, that you "seed" your SOA center of excellence. This is

11.3. Future Directions

As of this writing, most of the aspects, tools, methods, and standards described in this book are in a state of early adoption and, in some cases, have been defined and their usefulness proven.

In the areas of modeling, aspects of security, and service assets development, we will soon see new ideas that follow the thoughts we have outlined as being based on experiences from real-life projects of various customers. This shows that the principles of SOA are applicable to many solutions in many industries. However, to become more efficient and react quickly to business needs, there is still a market to establish of industry frameworks, common taxonomies, and semantics for specialized services and tools.

With the model of an on-demand operating environment in mind, you can expect to see special services to come that not only relate to business applications and integration for B2B or electronic markets, but also utility services that allow you to create an entire market of service providers on the Web. In addition, there are many ways to implement SOA that are agnostic of the programming platform, with various possible technologies you can choose to implement.

11.3.1. Technology Standards

Some standards have been well established for a while, thankfully, such as the SOAP messaging protocol and the WSDL description language. There are others, however, that are still at the proposal stage, while still others are starting to reach maturity within standardization organizations like OASIS and the W3C. We hesitate to indicate which proposed standards, due to the fairly rapid rate of progression of standards today. You should examine the Web sites of these organizations for their Web services proposed standards to ascertain their current status.

A.11.1

11.3.2. Web Services Monitoring and Visualization

Current research on Web services monitoring and visualization, including event-driven architectures that support these tasks, is starting to emerge from research labs. More sophisticated policy and management tools can help the SOA administrators and operation analysts to better care for the flexible, self-organization environment of SOA-enabled organizations that operate on a large, even global, scale.

11.3.3. Semantic Web Services

Semantic Web services are another topic that is gaining weight, especially when you have to deal with increasing numbers of connections, foreign units, and periodically emerging new approaches, products, and ideas that all need to be communicated and tested so that they can be used as designed. Semantic services provide a deeper understanding and provide context for how the information or the service is supposed to be used. Such semantic information enables architects and designers to know how to more effectively incorporate these services into their applications and architecture. A corollary is that new forms of directory and search engines will likely emerge from this activity.

11.3.4. Open Development Platforms

The development tools for SOA will be based on common, open platforms in which various individuals' roles can match the instrumentation and cooperate, just as the various SOA roles depend on each other. Open software development platforms, such as Eclipse, thus will become more integrated with business modeling and business process tools. This avoids or minimizes locking into proprietary development platforms, a significant obstacle particularly when engaging multiple organizations or companies in your SOA applications.

11.3.5. Services Assets

There is now an evolution toward support for model-driven development, and for finding and reusing service

11.4. Summary

As you can see, SOA can be a large subject rather than just one specific topic. It reaches across many existing problems that both business and IT have faced for a long time. Although it provides a paradigm of how the two worlds can work together, this does not come without significant effort. The promises of overall business agility and versatility may be the real motivators for this merger. As a departing thought, we would like to reiterate the true impact of this new paradigm: SOA can transform the very life structure of not just IT but the functions of the entire organization.

11.5. Links to developerWorks

A.11.1 You can see an up-to-date list of technology specifications and standards related to SOA and Web services at http://www-128.ibm.com/developerworks/webservices/standards/.

A.11.2 IBM's programming model for SOA enables software developers to create and reuse IT assets, using component types, wiring, templates, application adapters, uniform data representation and an enterprise service bus. http://www.ibm.com/developerworks/webservices/library/ws-soa-progmodel/index.html.

A.11.3 This three-part series by Alan Brown and Jim Conallen explores the nature of Model-Driven Architectures in depth. http://www.ibm.com/developerworks/rational/library/3100.html.

Glossary

access control list (ACL)

A mechanism for determining the access level and permissions that a given computing resource, such as a file or database field, provides for a given identity.

agility

The capability to lower the enterprise's center of gravity and move with suppleness, skill, and control. See also business agility.

application programming interface (API)

A set of routines and function definitions that abstract the implementation details and make it easier to develop and build software applications.

application server

An application server is a server-side program in a distributed network that is dedicated to hosting the enterprise application's business logic. It provides the middleware infrastructure as part of a multitier application, consisting of a user interface server, a business logic server, and a database or transaction server.

Association for Cooperative Operations Research and Development (ACORD)

A global, nonprofit insurance association whose mission is to facilitate the development and use of standards for the insurance, reinsurance, and related financial services industries.

authentication

The validation and verification of the identity of a user, device, or some other computing entity, often as a prerequisite to allowing access to resources in a system. Authentication merely ensures that the entity is who it claims to be, but it says nothing about the access rights of the entity.

authorization

The process of granting or denying access to an individual or computing entity. This allows access to various resources based on the entity's identity. See also access control list(ACL).

business agility

The capability of an enterprise to respond with speed to market opportunities, external threats, or customer demands by changing its business processes that are integrated end-to-end across the company and with key partners, suppliers, and customers.

business process

A set of logically related tasks performed to achieve a defined business outcome. A process is a structured, measured set of activities designed to meet the business objectives.

Business Process Execution Language (BPEL)

Index

Index

Index

Index

Index

Index

Index

Index

Index

Index

Index

Index

Index

Index

Index

Index

Index

Index

Index

[A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X]

Index

Index

[A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W] [X]

Index

Index

Index

Index