

Bringing SOA to Life: The Art and Science of Service Discovery and Design

Practical guidelines and experiences from real-world SOA projects

■ In a service-oriented architecture (SOA), a service is a unit of work performed by a service provider to achieve desired results for one or more service consumers. A service provides a function that is well defined, self-contained (for example, loosely coupled to its environment), described solely by its interface contract and behavioral attributes (for example, it hides implementation), and located anywhere on the network.

Essential components of an SOA – such as a service provider, service repository, service mediator, and service consumers – all rely on service definitions as the key element to describe, access, transport, and understand services.

Modern SOA as a pattern for transforming enterprise architecture is still emerging, as are the strategies for service discovery and design. At this juncture of SOA's maturation, it's highly useful to consider concepts and practical experience gleaned from serious and substantial SOA implementations. In this article we provide both. For various aspects of service design activities, we discuss general considerations that are often motivated by well-accepted

WRITTEN BY
MANAS DEB,
JOHANNES HELBIG,
MANFRED KROLL,
& **ALEXANDER SCHERDIN**

software engineering ideas. We also describe the experience of Deutsche Post, Germany – one of the world's largest global logistics service providers – where SOA concepts have been widely adopted at the enterprise level.

SOA at Deutsche Post is business focused. Several SOA-specific processes and methods have been established to put the promised benefits into practice. Further, a highly sophisticated, enterprise-quality service backplane called SOPware (service-oriented platform) that leverages Oracle's Fusion Middleware Suite provides the supporting infrastructure for business-service mediation. Thus, business units don't have to worry about the technical details of service implementation and provisioning; instead, they can quickly and easily assemble

high-level business services and processes.

This article focuses on the crucial aspects of bringing SOA to life: how to establish business ownership, how to discover and design services, and how to foster an SOA-based enterprise architecture transformation.

The Starting Point: Service Discovery and Portfolio Management

Business Services vs Technical Services

The principal attraction of SOA is its ability to drive IT evolution through business *and* to enable better business operations through a more flexible IT. While the basic concepts of SOA can be helpful to build more efficient IT operations, the larger benefits of SOA are obtained when business operations can quickly build competitive business applications and make such applications resilient to change. Consequently, services used by high-level business processes, which we call business services, are the main focus of this article. Technical services, in contrast, are typically embedded in the infrastructure that supports these business services. As important as these technical services are, a good SOA implementation should clearly separate business from technical services.

To realize the full transformation potential of an SOA, business people must take responsibility for identifying business services and

describing their characteristics. To that end, several enterprises, including Deutsche Post, are using the concept of business domains to establish business ownership.

Establishing Business Ownership for SOA

To create a common language for business and IT people to discuss potential services and to foster ownership for specific business services, it's wise to look at the fundamental structure and relationships defining the business (for example, a customer buys a product, or an invoice is sent to a customer). Here, the scope of an organization's business is first factored into business domains, which are essentially blocks containing closely related functionality and data, such as customers, products, and contracts. In order to design a domain landscape that renders the benefits of long-term stability and thus provides solid ground for introducing SOA at all levels, it's important to understand how a specific enterprise conducts its business (see Figure 1).

Practical experience at Deutsche Post shows that identifying core business objects and their relationships provides a suitable first iteration of constructing the domain architecture. Deeper analysis of the business processes and their interactions will lead to more precise domain descriptions, as well as the definition of subdomains for some major domains. We strongly recommend checking the robustness of the domain architecture by mapping it against actual and potential future business scenarios.

Designing business domains must be based on deep business understanding, but it also requires some gut feeling. Let's consider two domains: customer and customer relationship. The first mainly manages key customer data required by most other domains, while the second tracks various aspects of the ongoing customer relationship. These domains may seem strongly related at first, particularly because both deal with customers and the domain names sound similar – so why not combine them into one? A close inspection of the fundamental business relationship reveals crucial differences. The customer domain provides information that changes relatively infrequently, but it needs to provide a 360° view of all aspects of a customer and is used in a similar fashion in most domains. The customer relationship domain, on the other hand, provides functionality that has a broad variety and changes frequently – as often as the

customer has some dealing with the company and as the CRM strategy of the company evolves. Business people responsible for customer relationships often use the information quite differently – for instance, some use it for real-time cross-selling while others use it to render better customer service. Thus, choosing separate domains for customer and customer relationship is well justified. (For other examples of domain architecture in mobile telecom and in banking,

see the fourth and fifth entries in the References section.)

This example points to another important consideration: to use SOA to transform an enterprise IT landscape, business people must take ownership of business architecture. Business owners are responsible for the scope of the domains and for providing ideas for those business services that their domains provide, as well as for identifying requirements for services

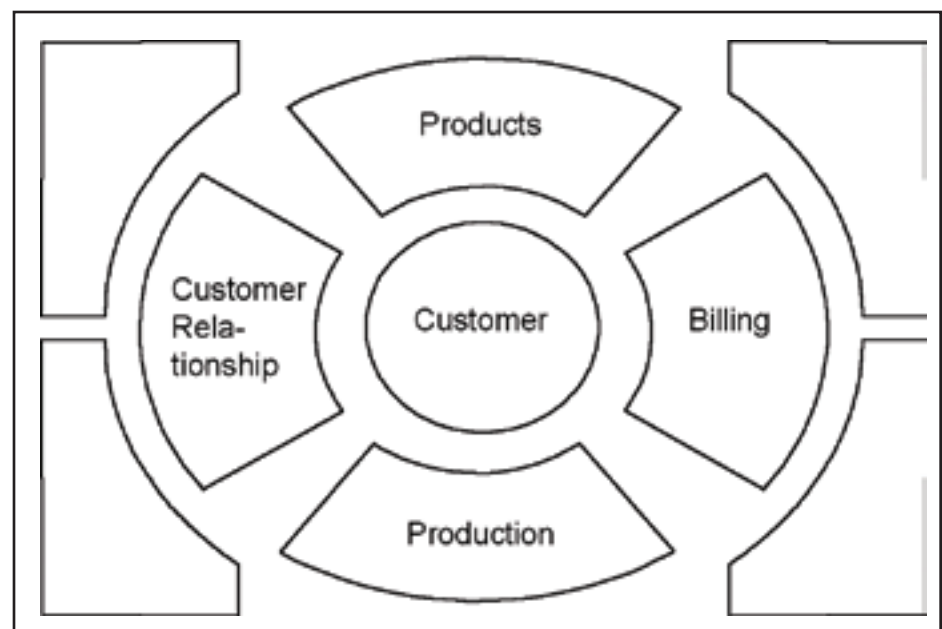


FIGURE 1 Simplified high-level domains of a logical service architecture

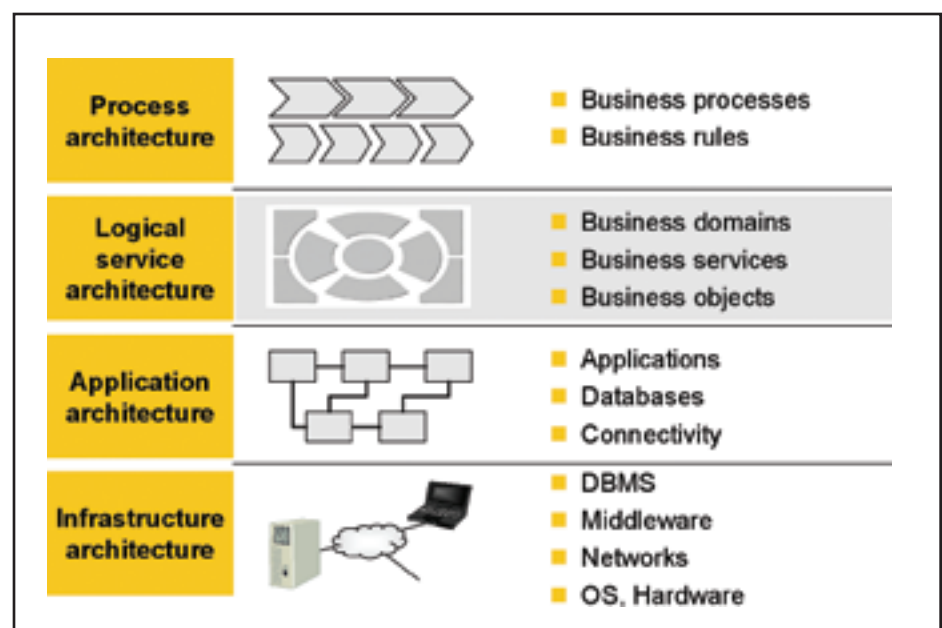


FIGURE 2 Logical service architecture as an abstraction layer between process and application architecture

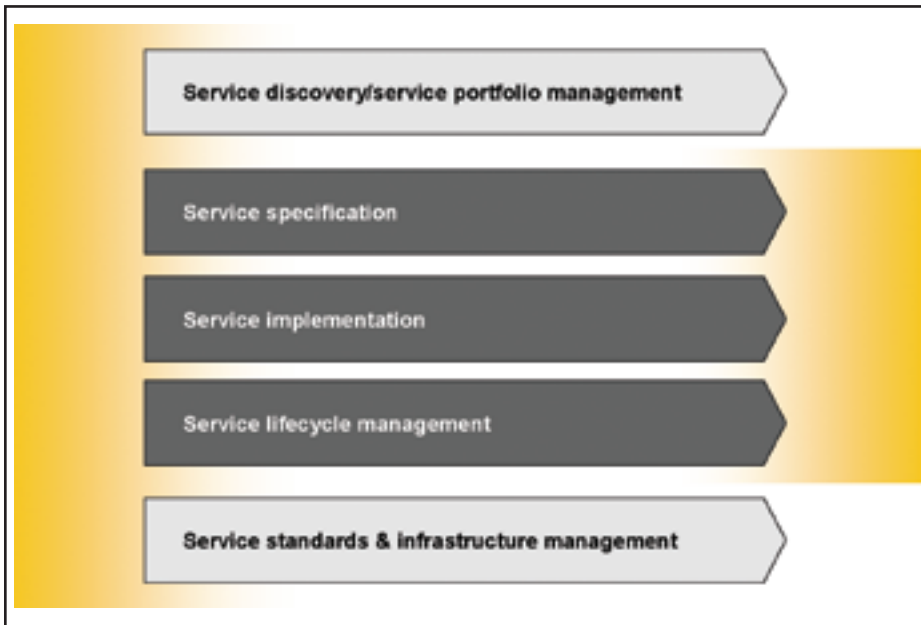


FIGURE 3 | Service design processes

that their domain needs (which are, in turn, provided by other domains). This establishes a federal business-IT governance, including clear definitions of data ownership. At Deutsche Post, enterprise architects and service designers from the business-IT organizations help the business owners work out the details; however, the final responsibility lies with the business side (see Figure 2).

The business domains, as well as the business services, establish an abstraction layer between the process architecture and the application architecture of an enterprise.

The benefit is clear: by decoupling processes and applications through domains and their associated services, change cycles can be managed independently. IT systems (both on the application and the infrastructure layer) may be changed or replaced without affecting business processes, as long as the service contracts are kept stable. Conversely, business processes can be changed or augmented relatively easily when many of the building blocks (the business services) already exist, waiting to be reused.

Service Discovery

While in some ways services are similar to components, they are also similar to mini-applications that must be managed in a similar fashion to applications. As shown in Figure 3, Deutsche Post defined a set of five service

design processes that provide a grid for creating and evolving the service portfolio – these processes have general applicability in any industry.

The first step of creating and managing a business service portfolio is service discovery. In order to look for a candidate set of services that a domain should provide, you can investigate the ensemble of business objects and their relationships. Business objects (such as customers, invoices, and addresses) are *not* IT objects, but entities required to do business described solely in business terms without reference to the specifics of any IT system. For example, one of the key services of the domain customer is likely to be CustomerInformation, and the elementary service operations associated with this business service are likely

to be CRUD (create, read, update, delete) type. However, more complex functionality that combines several basic service operations, may also be associated with this domain. For example, a customer domain may host a CustomerAddressConsistency service, encapsulating the check for existence and proper name of the customer, as well as verifying (and potentially rectifying) her address. Other interesting service candidates may be found by looking at specific combinations of key business objects (such as looking at contracts and their associated invoices).

This top-down approach to service discovery can be complemented by two other approaches. Tracing business processes, following their key interactions with their business environment, and grouping similar functionality can lead to service candidates potentially overlooked by a pure top-down approach. This business process-driven approach can provide a sanity check for completeness and give a first indication of the reuse potential of specific services. However, it requires some degree of caution: different business processes (potentially defined by different business people) often feature different semantics for the same (or similar) concepts. Our experience shows that a thorough and rigid analysis of business semantics (including translating or even harmonizing business terms) is a key prerequisite to achieving a working service portfolio. Note that this approach alone may easily lead to a too fine-grained definition of services.

Additionally, a usage-driven approach to service discovery looks for which IT infrastructure (such as mainframe or legacy) application functionalities and data have been used by business applications to date, wraps service interfaces on them, and uses these services

“ As important as these technical services are, a good SOA implementation should clearly separate business from technical services ”

for future business applications. While this so-called bottom-up approach may complement the two other approaches for service discovery, we believe that the lack of semantic harmonization of services will not lead to a sustainable SOA transformation. We'd recommend this approach only as a starting point – for example, to prove the general working of an SOA in a political environment.

Further, we suggest starting service discovery with a small investment in creating conceptual services (for example, only specification but no realization) that combines the aforementioned approaches, and start with actual service implementation in a step-wise fashion, strictly based on business priorities and business cases for individual projects. Business analysts and enterprise business architects obviously play a major role in the business service discovery phase.

Making Change Happen: Enterprise Architecture Transformation Through SOA

Building the Service Portfolio

At Deutsche Post, service discovery yielded a portfolio of nearly 100 business service candidates (each, on average, featuring 5 to 10 service operations) that were subsequently prioritized by factors such as business value, reuse potential, and IT complexity reduction potential. These candidates provide the baseline for an ongoing service portfolio management process that refines the roster of candidates and triggers detailed specification, implementation, reuse, and life-cycle management of specific services.

There are many important issues to deal with while building up a service portfolio, such as service ownership, funding for creating and maintaining services, and life-cycle management. Based on the balance between focusing on SOA services and focusing on project execution, different service acquisition styles can be recognized. For example, one style might undertake an enterprise-wide effort to create or acquire most of the potentially useful services and then start a series of SOA projects that would implement (and use) them. This more services-focused approach aims at first creating a rich service portfolio but involves solid, long-term strategy and some degree of up-front investment (most notably for service

design methodology, as well as for a service mediation platform). Another, more projects-focused approach would give priority to building services as required by some projects without much emphasis on how they might be shared. This approach would provide short-term returns but could hamper long-term SOA benefits.

From the beginning, Deutsche Post has emphasized an evolutionary approach for architecture transformation in accordance with SOA principles. This managed evolution approach aims to define smaller IT projects with positive business cases and limited scope,

by a project. Service specification at Deutsche Post has three steps. The first step is taken during the service discovery phase, rendering an *enterprise service description* in the context of Deutsche Post's enterprise architecture model. This enterprise business object and service model describes the logical service architecture view and contains domains, business objects, and services. The services are described as interfaces in this model (UML is used as a formal language), while the description of service operations refers to business objects and other data types in the model.

The second step, the detailed *business ser-*

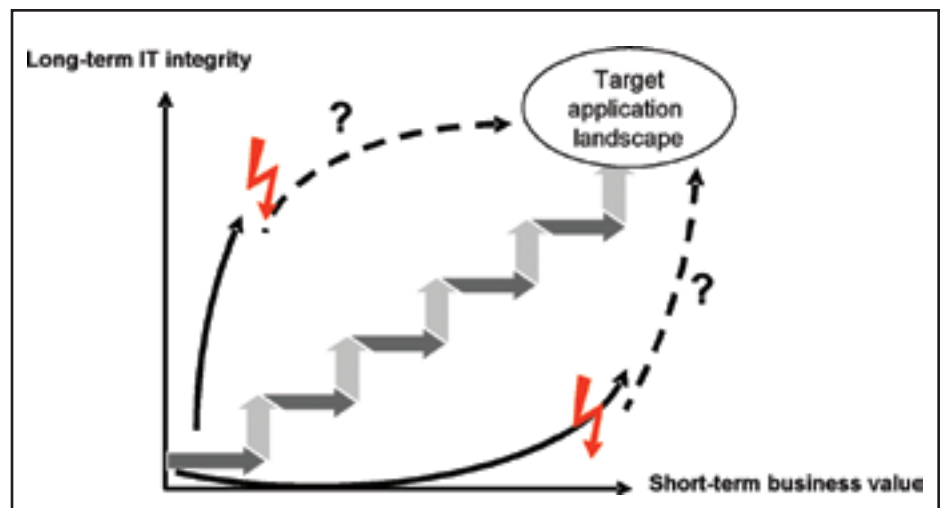


FIGURE 4 SOA adoption through managed evolution

so as to provide short-term business benefits (tactical aspect) while complying with a target application landscape (the SOA blueprint along the business domains) defined at the enterprise level. These projects must have a business owner and will be conducted anyway, with or without an SOA. However, by identifying the services that could be either implemented or reused by a project, the managed evolution approach creates a desirable balance between the services-focused and the projects-focused approach. Each project will thus create business value *and* contribute to the evolution of a flexible IT landscape based on a broad service portfolio (see Figure 4).

Specifying Services

Additional refinements pertaining to various characteristics of this service are required for each candidate service to be implemented

vice specification, includes refining the service interface and taking specific aspects of the IT project (such as potential limitations of the service scope) and then realizing or acquiring the service under consideration. Business service specification is critical in the service design processes, as these specifications are the essential communication method for all parties interested in the service. You should consider the following key categories of information while creating service specifications:

- **Definition:** Service and service operation names, input and output parameters for each operation, admissible invocation styles (such as synchronous or asynchronous), and service classification (such as business, technical, or infrastructure; or process-oriented, business logic, or data access).
- **QoS/SLA:** This category has information on the service's performance quality, including

useful details on load versus response times, availability and fail-over responses, and possible SLAs.

- **Constraints and Policies:** This includes pre and post conditions relevant to the service invocation, admissible/expected ranges of input/output parameters, including error returns, and particular sequences in which operations should be invoked. Security guidelines and other essential policies can also be included here.
- **Ownership:** Information on those who own or maintain the service.
- **Release Information:** This groups status or planning information about past, current, and planned versions of the service, plus information on any interversion compatibility-related issues.
- **Operational Behavior and Usage Patterns:** This category, which is particularly helpful for complex or nonobvious services, should provide information about different and recommended ways to use the service.

The service specification exercise is completed in the last step, the *technical service specification*, where service mediation platform-specific requirements are spelled out, and other missing technical requirements (such as specific throughput or message sizes) are defined in detail. Proper classification of a service also helps you apply design and implementation best practices to that service.

The functionality, quality, and policies that govern the service render an agreement between the provider and the consumer, the so-called service contract. Currently there are no industry-wide accepted standards for service specification. However, WSDL and UDDI are becoming the de facto standards for describing and publishing the basic service definition. Tests using Web Services Interoperability (WS-I) specifications are also becoming popular with services aimed at wider reuse.

Specification iterations produce finer and more concrete details about the service interfaces and functionalities, thus making the service more flexible and cost-effective by incorporating present requirements as well as projected future needs. Service specification iterations typically require close collaboration between business service owners, business analysts, and technical analysts.

Optimal Service Granularity: Fine or Coarse?

A popular debate in the SOA community concerns how fine or coarse the services should be. While there's no standard way to quantify service granularity, we can use some ideas from component-based design, such as the number of function points or data elements affected by the invocation of a service. If a service needs to be called

too many times in a business application or if only a small part of its functionality is typically used, it's likely that the service is too coarse. If too many parameters for a service are required, the service is most likely too low level and fine grained. While these observations are based on real-life service implementations, in our experience this question is resolved by effective service portfolio and life-cycle management processes: the degree of appropriateness of a service portfolio grows steeply over time by starting with high-value service operations and expanding and evolving them over time. Striving for an appropriate granularity will maximize ease of use, reuse, and manageability; an appropriate service is not necessarily either fine or coarse, but one that maximizes business value.

At Deutsche Post, mostly coarse-grained services were implemented at first. These were a rather stable set of fundamental services that could be initially established. Over time, the number of service consumers grew and more specific requirements emerged; thus, more fine-grained service operations are currently implemented. These elementary (or atomic) service operations provide a baseline for defining a rich portfolio of more complex service operations – for example, by combination (compound service operations) or by process orchestration (orchestrated service operations).

Service Implementation and Life-Cycle Management

While in most ways service implementation is not much different from software development, we'd like to point out two important differences.

First, service interface code for a chosen service mediation platform doesn't need to be hand coded, but it can – with a capable service-design tool chain – be generated. Deutsche Post uses a service-design tool chain based on the model-driven architecture paradigm. The starting point is the enterprise business object and service model, which is transformed to a project business object and service model (PBSM). After the PBSM is further refined as part of a project specification, the tool chain generates both code skeletons for Deutsche Post's SOPware service mediation platform and service specification documents. While the service specification artifacts are modeled with appropriate tools, XML and XMI are chosen as proven and standardized



WEEKEND WITH EXPERTS

Get a Java injection on the go!

Spend a weekend and have a lunch with experts in an intimate learning environment in the city near you.

No salesmen allowed. Join our technical discussions and hands-on workshops.

The next Roadshow is in Edison, NJ on March 11 and 12.

www.weekendwithexperts.com

exchange languages.

Second, when setting up or acquiring a service mediation platform, one must consider support for service testing, since services by definition require a loosely coupled environment. For example, at Deutsche Post, service module tests use the DevBox, a component of the SOPware that provides service invocation libraries and simulates service providers on a developer's local machine. Additionally, Deutsche Post has set up a service architecture test lab that provides an environment that comprises all services and service providers in order to integration-test SOA applications.

While service interfaces will be stable elements of an enterprise architecture, services will grow (in terms of additional service operations added over time) and service implementations will change as service providers behind the façade change. While an SOA minimizes the impact of this change, the life cycle of services must still be managed. The elements of service life-cycle management are version control, a business service repository (including service models and specifications), and a technical service repository (service registry) that enables physical access to services at implementation and run time.

Technical Considerations: Service Platform and Standards

As we've seen above, business services can effectively decouple business change and IT change. However, what if the IT infrastructure (or elements of it) that mediates the services needs to change? We've seen enterprises entangled in an EAI or middleware legacy, unable to change at the speed required by the business. When some of those technologies became obsolete or couldn't cope with growing performance requirements, tool-specific EAI adapters and interfaces based on proprietary middleware protocols couldn't be migrated with reasonable effort. The promise of flexibility through technology proved to be futile.

Thus, we recommend a standards-based approach for service mediation, such as the new Java Standard Java Business Integration (JBI) or an approach similar to Deutsche Post's SOPware (using, among other standards, JBI). In SOPware, another abstraction layer has been set up, this time with the purpose of decoupling business service logic from the underlying IT infrastruc-

ture. This layer is purely based on standards such as J2EE (.NET would be an option, too), XML, and WS-I. Driven by technological progress and functional expansion, SOPware, created in 2001, effectively replaced or augmented nearly all major infrastructure components (such as application server, MOM, directory server, and transformation engine) without affecting the developer interface. Simple XML-based APIs to create and invoke services are the only aspect of technology that developers need to know about service mediation, while the specifics of underlying technologies and tools are hidden. Programmers can thus focus on implementing business logic, rather than on low value-add infrastructure code. This strategy, for Deutsche Post, provided the ultimate flexibility and investment protection – not just toward business logic, but also toward IT infrastructure.

Conclusion

While many scientific principles can guide service discovery and design, as with any architectural work, some gut-level experience from industry veterans can go a long way in the process – particularly in the early iteration cycles. Making SOA a reality calls for quick and easy implementations: business people should focus mainly on the business logic, not on protocols and transport. Thus, a full-featured SOA backplane can be really useful. Also, using open standards and a modular suite of technology components should promote higher reusability of SOA assets at lower TCO, minimize vendor lock-in, and reduce SOA adoption risks.

Put into a short equation, SOA = semantic integration + loose coupling + managed evolution. However, our most important message is to use SOA as a common language between business and IT people, thus bridging the gap that has, for a long time and at many enterprises, blocked the path to flexibility of both business processes and the IT landscapes that support them.

References

- SOP management paper (Deutsche Post World Net – SOP Group), 2004.
- SOP SBB technical paper (Deutsche Post World Net – SOP Group), 2004.
- SOA Is More Than Hype (Johannes Helbig), Presentation at Gartner Application & Web Services Summit, June 2005, Barcelona.
- Reusing IT Components in Mobile-Telecom

Companies (Enrico Benni, Klemens Hjartar, Jürgen Laartz, and Alexander Scherdin), McKinsey on IT, Fall 2003.

- Services, Events, and Contracts – SOA at Credit Suisse (Claus Hagen), Presentation at SOA Days, February 2005, Bonn.
- Designing IT for Business (Jürgen Laartz, Eric Monnoyer, and Alexander Scherdin), McKinsey Quarterly, 2003, Number 3.
- SOA Is More Than Hype – Use It for IT Business Integration (Mark Happner, Michael Herr, and Alexander Scherdin), Presentation at SOA Days, February 2005, Bonn.
- SOA – Complexity Management Through Integration (Johannes Helbig), Presentation at OOP Conference, January 2004, Munich. ■

About the Author

Dr. Manas Deb, a senior director at Oracle's Fusion Middleware Group, currently leads strategic engagements operations for Oracle's service-oriented integration solutions. He has worked in the software industry for nearly 20 years, half of which he spent architecting and leading a wide variety of enterprise-level application and business integrations projects. Manas has a PhD in Computer Science and Applied Mathematics, as well as an MBA. ■ ■ ■ manas.deb@oracle.com

Dr. Johannes Helbig is considered among the first to have formulated the concept of a service-oriented architecture, and is father of the SOA program at Deutsche Post, where he currently serves as member of the board and CIO of the MAIL division. He holds a doctoral degree in theoretical computer science and his main interests include IT architecture and IT management.

Manfred Kroll is director of Business Architecture and Processes at Deutsche Post's MAIL division. As chief enterprise architect, his objective is to implement a cooperative, SOA-based application landscape pursuing an evolutionary approach. Before joining Deutsche Post MAIL, he held several management positions, most notably at IBM Development Labs and at T-Mobile. Kroll has a Masters degree in Computer Science from University Dortmund, Germany. ■ ■ ■ manfred.kroll@deutschepost.de

Dr. Alexander Scherdin, senior professional for IT Service Design and SOA at Deutsche Post's MAIL division, is responsible for broadening the company's service portfolio and driving forward the definition and execution of its service design processes. Before joining Deutsche Post MAIL, he worked as an IT architecture consultant at McKinsey & Company's Business Technology office. Scherdin studied at the University of Frankfurt, Germany, and holds a PhD in Theoretical Physics. ■ ■ ■ alexander.scherdin@deutschepost.de