SECTION ONE:

# WHAT IS SOA AND WHY SHOULD I CARE?



If you haven't heard about *service oriented architecture (SOA)*, then you have likely been living under a rock for the past several years (it is also very unlikely that you would purchase a book on the topic). There is a tremendous degree of hype, FUD (fear, uncertainty, and doubt), and misinformation floating around regarding SOA, service orientation in general, and what it really means for modern enterprises. This first part of the book is aimed at cutting through all of this and providing a solid foundation in SOA, its huge potential, and its inherent risks.

Chapter 1, "SOA Primer," introduces and defines SOA, explains what it means to be service oriented, and describes how we evolved to this point. The chapter introduces the typical architectural layers that comprise an SOA enterprise solution and the key SOA infrastructure elements that are commonly found.

Chapter 2, ''Business Process Management and SOA,'' introduces and defines business process management (BPM), explains what it means to be process-centric, and describes how all of this relates to SOA. Alignment between IT and business through BPM is examined, along with the relationship between objects, services, and processes. Finally, process modeling is explored in great detail.

Chapter 3, ''SOA Value Proposition,'' identifies the four core SOA value propositions (reduced integration expense, increased asset reuse, business agility, reduced risk) as well as several emerging values (alignment, time to market, visibility, and modernization). These value propositions are then explored by looking at the two fictitious case studies used throughout this book.

Chapter 4, ''Risks in SOA Adoption,'' takes a raw and honest look at IT challenges and barriers to SOA success. Common SOA promises are examined, including business and IT alignment, process automation through SOA, service reuse, service composition like LEGO® blocks, smoother integration through open standards, and improved business responsiveness. SOA has potential, but this chapter provides a very real look at the risks inherent within SOA.

CHAPTER 1

# SOA PRIMER

You awake to the familiar buzz of your alarm clock and stumble out of bed and into the bathroom. With a flick of a light switch you are blinded by the bathroom light (unless you have one of those fancy bathroom lights that gradually brightens to allow your eyes to adjust). Later you plug in your coffee grinder, grind some fresh beans, and then brew a steaming pot of coffee. Throughout your morning routine, you use electricity. You use as much or as little of it as you need and you do so with little regard for how much electricity you have consumed that day, week, or month. Some weeks or months, travel and work schedule may dictate less time at home (and less electricity consumption); other days or weeks, you may consume much more. Electricity is a service. It is available on-demand based on a predetermined fee structure and is delivered consistently based on industry standards and regulated infrastructure. Electricity, like other utilities, is service oriented.

## FROM AD-HOC SOLUTIONS TO SERVICE ORIENTED CAPABILITIES

At first glance, *service oriented architecture (SOA)* sounds like a techie thing with little relevance to business and delivering customer value. But service orientation is more than just a technical architecture; it is a movement within government organizations and private industry that is transforming business value chains, organizational alignment, and technical delivery capabilities.

To better understand this transition, we will first examine the evolution within the electric utility industry from ad-hoc creation of electricity toward a true service oriented model. Then we will explore the parallels currently occurring within the realms of business and technology with respect to SOA.

### Edison Had a Neat Idea

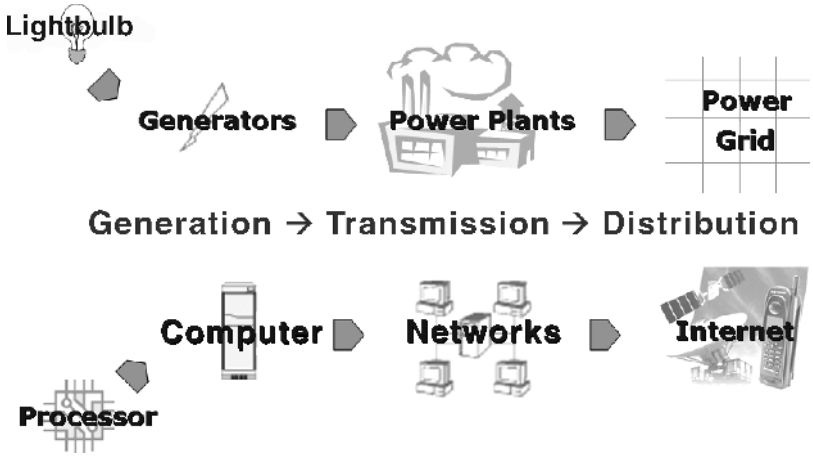
Generating electricity to illuminate a bulb is a pretty cool concept. The means of getting the electricity to the bulb has evolved over time. Creating that electricity via generators was a fine initial implementation, but that method was not as economical or reliable as desired. Generators required individuals and businesses to stockpile fuel in order to produce electricity. They also had limited ability to regulate the electricity flow, resulting in reliability problems as well as safety concerns. Later, the electricity needs of towns and cities were supported by power plants. Generation of power within homes and businesses gave way to transmission of power from centralized plants via electrical lines. Eventually, these plants connected with one another via a standardized power grid, enabling the exchange of power supply across great distances. Power demand could now be supplied by plants in other regions via the power grid. As demand changes, individual plants can throttle the supply of power, enabling the entire grid to respond to market needs.

There is another interesting aspect to the electric power industry, and that is the *economics of deregulation*. Although in some parts of the world electric service is owned or at least heavily regulated by the government, others have deregulated and embraced a free-market model. In these deregulated markets, private industry can build a plant, generate electricity, connect to the grid, and negotiate service levels and a price to sell this electricity to brokers. Industry standards, transmission protocols, and robust infrastructure enable a truly service oriented industry in which demand can wax and wane, supply can be delivered from anywhere on the grid, and new providers can enter the market and negotiate price and service level agreements (SLAs) as needed.

**Service Orienting Modern Enterprises Is a Good Idea, Too**

From localized generation of electricity to transmission of electricity from centralized power plants to distribution of electricity via a power grid, the electric utility industry has evolved into a service oriented model. As illustrated in Exhibit 1.1, this same evolution is taking place in modern enterprises today. Originally, businesses deployed local software (applications and databases) and hardware (personal computers and servers) to support business operations. Large, distributed businesses would require multiple instances of such software. Later, network infrastructure and distributed computing technologies allowed businesses to deploy centralized solutions (software and hardware) with distributed client-side access in lieu of multiple copies of the full software/hardware stack. These centralized solutions are much more economical and more powerful than having a bunch of solutions deployed in every location. The drawback, however, is that these solutions are not flexible. They offer a monolithic, one-size-fits-all solution. If you need to tweak one aspect of business operations (e.g., modify your

EXHIBIT **1.1** *As with the electric industry, the computing industry has evolved into a service oriented model*



supply chain process, change the data processing logic for one product type, outsource one component of the application, etc.), you generally have to go through a long design–development–testing–deployment life cycle. Service orientation is about taking those monolithic solutions and breaking them up into flexible, reusable, and configurable components. These components, or services, are available to service requests from anywhere in the network without the traditional barriers of operating system, programming language, or platform technology. Additionally, these can be reconfigured and a chain of services rearranged in a fraction of the time that traditional solutions can be changed in order to respond to changing business needs. To return to our electric utility industry analogy, service orientation allows enterprises to respond more readily to electricity demand (service requests) and to adjust power supplied by power plants (reconfigure service providers) to adjust to the demands of the grid (network).

Finally, there is the issue of economics and deregulation. Just as a deregulated power industry permits new providers to join the grid and sell power to customers, so, too, does a service oriented enterprise model. The key in both cases is industry standards, transmission protocols, and robust infrastructure. By service orienting the enterprise, businesses introduce the potential to connect systems and databases within their internal enterprise and even connect to trusted partners and third-party service providers. Why maintain an address cleanup capability when you can simply invoke address services maintained by the U.S. Postal Service (or similar national postal service)? Why maintain your own geographical tracking and management capabilities when you can simply call services made available by Google Maps? Service orientation allows business needs to be fulfilled by any provider within the local or extended network, provided that they support the appropriate technology standards, message transmission protocols, and required SLAs. On-demand, service oriented capabilities, backed by service contracts and enforceable SLAs—imagine scaling your business, meeting increasing customer demands, and doing so as effortlessly as you turn on a light bulb.

## WHAT EXACTLY IS SOA?

In exploring SOA, we will start by defining the concept and then look at some of the most common components that comprise SOA solutions.

### Defining SOA

SOA can be expressed very simply:

> SOA is about connecting customer requirements with enterprise capabilities, regardless of technology landscape or arbitrary organizational boundaries.

Digging in further, we learn that SOA means different things to different people. At a very low level, it is a technical architecture supported by standard formats and protocols. At a more general level, it represents a shift within the enterprise toward breaking up organizational silos and monolithic information systems to enable flexibility in how customer solutions are assembled. Chiefly, SOA aims to align technology investments and initiatives with business goals through an enterprise governance plan.

In some respects, the *A* in *SOA* is a bit unfortunate. While architecture is certainly a key aspect of any successful SOA initiative, it tends to give the erroneous impression that SOA is an "IT thing" that the business community need not worry about. The reality is that service orientation is an enterprise strategy with far-reaching implications into business capabilities, organization structure, technical infrastructure, and the overall agility and efficiency of enterprise operations. Consequently, a distinction will be made in this text between SOA (a style of enterprise architecture) and service orientation (an enterprise strategy that focuses on business processes, serving customers, and alignment of enterprise resources with business objectives).

## DECONSTRUCTING SOA

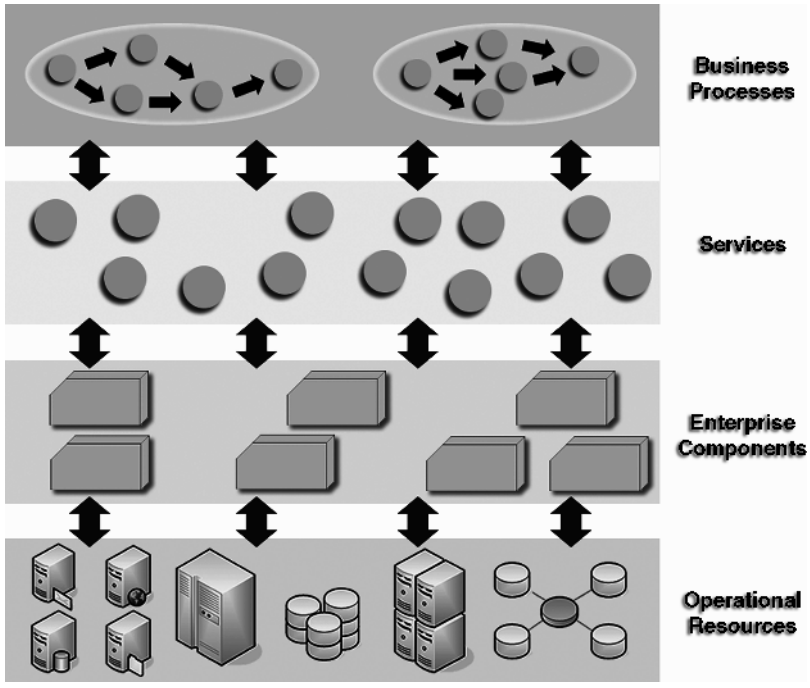
No two service oriented enterprise architectures look the same. SOA is an architectural style with a handful of common elements and themes and myriad implementation strategies. A nominal, representative

architecture can be identified in order to better understand SOA and "what it looks like." A reference diagram depicting the SOA layers is illustrated in Exhibit 1.2. This diagram will serve as a useful reference in this section and throughout the rest of the book. While any given implementation of SOA may be more or less complex than this model, this diagram provides a good starting place.

The layers illustrated in Exhibit 1.2 are as follows:[1]

- **Operational resources.** Comprised of existing systems, applications, and databases, the operational resources layer represents the legacy enterprise. Your customer relationship management (CRM), enterprise resource planning (ERP), and product life-cycle management (PLM) systems are good examples of operational resources. Some of these systems are commercial off-the-shelf

EXHIBIT **1.2** *SOA architectural layers*



(COTS), while others are homegrown, but all of them house valuable enterprise data and business logic. The services that are made available through an SOA leverage these existing investments and uncover new opportunities for utilizing these assets within a larger enterprise context.

- **Enterprise components.** Sitting atop the operational resources is a layer of enterprise components. Enterprise components typically employ container-based technologies such as CORBA, EJB, COM, DCOM, .NET, and the like. These assets are responsible for managing custom business logic and interfacing with the operational resource layer to carry out this logic. Additionally, they support the scalability and quality-of-service requirements of the services exposed in the layer above. A service's ability to support contracted SLAs is based on how well designed the enterprise component layer is that supports the service.
- **Services.** Capabilities from the enterprise component layer are selectively identified as services. The analysis, design, and development of these services is then funded and the services are deployed in order to expose these capabilities through well-defined interfaces. Service descriptions, quality-of-service (QoS) SLAs, and other key service metadata are also defined to accompany these important SOA assets.
- **Business processes.** Individual services provide incremental value for an organization but will likely never transform the way business gets done. Business processes, however, represent powerful orchestrations of one or more services that solve a business problem. Services are bundled together into a logical flow (described as *orchestration* or *choreography*) to solve some sort of end-to-end business problem. For example, one service might provide access into the purchase order mechanism for an ERP system and another provide access into customer account capabilities within the CRM system, but a business process could lump these services and perhaps others together in order to complete an order fulfillment request.

### Key Infrastructure Elements

Just as every SOA is likely to be different, the infrastructure that enables that architecture will also vary. There are, however, some common components and SOA infrastructure pieces that you are likely to encounter when exploring SOA enterprise solutions. These include:

- **Business rules engine.** This allows business logic to be defined in such a way as to enable business owners (especially the line of business managers) to tweak and throttle key variables that drive certain business processes. Examples include tweaking insurability thresholds in the insurance industry and throttling service performance to respond to increasing seasonal demand in the retail industry.
- **Enterprise Service Bus (ESB).** Considered by some to be the quintessential SOA infrastructure element, an ESB can be used to broker service transactions, map interfaces and data sets (enabling clients and services with differing expectations to communicate seamlessly), route traffic to appropriate services based on internal logic, and perform other value-added service-brokering solutions.
- **Policy server.** Governing SOA to ensure that business objectives are met and that the enterprise is not exposed to undue risk is crucial. One mechanism for governing SOA is through the definition and implementation of policies, which are then applied to business processes as well as individual services. Policies will be discussed at length in Chapter 11, but essentially represent declarations regarding the use of service data and metadata or other nonfunctional qualities such as performance, security, or service reliability.
- **Service container.** This is where the services actually live. Resource pooling and intelligent caching may be implanted here to improve performance. This is typically some sort of application server and may, in fact, be bundled into an ESB platform.
- **Process engine.** This supports the definition, configuration, and execution of business processes (service orchestrations), and manages these processes and invokes service operations to fulfill process

activities in a well-defined sequence. It may exist as a standalone installation or be bundled within an ESB platform.

- **Service manager.** The service manager is responsible for service life-cycle management, monitoring service health and performance, client access tracking, and in some cases even enforcing policies and SLAs. The service manager may also manage service versioning. Finally, it might exist as a standalone installation or be bundled with a service container, a policy server, an ESB platform, or any combination thereof.
- **Service registry/repository.** With few exceptions, this infrastructure element will exist for every SOA enterprise. Depending on the size and requirements of the enterprise, any of the previously identified infrastructure elements may or may not exist. The registry/repository is crucial, because it serves as the directory for service descriptions, interfaces, and other key metadata. Services also can be organized within the registry/repository according to a predefined or organization-specific taxonomy or categorization schemes to support service discovery. Some registries/repositories are deployed independently, while others are bundled with a service manager, policy manager, ESB, or some combination thereof.

## IS SOA THE LATEST INDUSTRY FAD?

The pace of change within the business community, and information technology in particular, rightly leads the savvy professional to question whether SOA is merely a fad. Technologies and trends come and go, so what makes SOA any different? Several factors point to SOA's longevity.

### SOA Is a Natural Evolution

To start with, service orientation evolved out of mature application and integration efforts in the late 1990s, and came on the scene around 2000–2001. Since that time, the adoption of Web Services and service orientation among vendors and private industry has been tremendous

(some research pegs the number as high as 90% among Fortune 500s). Federal and state governments are even engaging in early service oriented initiatives. Virtually every vendor of enterprise systems now has an SOA initiative to one degree or another. Some enterprises are able to jumpstart their SOA efforts merely by upgrading to the latest releases for their major COTS systems. Even CICS mainframes have gotten on the bandwagon. The latest version of CICS includes native support for Web Services. This is, in fact, the trend throughout the industry.

### SOA Has Staying Power

All indicators point to SOA remaining a viable and lasting part of the enterprise. Consider the following quote from Gartner in a November 2006 research note:[2]

> SOA will be a *durable change* in application architecture, more like the relational data model than shorter-lived concepts, such as distributed object computing using object request brokers.

By placing service orientation alongside the other major shifts in information technology (IT) (see Exhibit 1.3[3]), the significance of its impact is made even clearer.

EXHIBIT **1.3** *Service orientation represents a major shift in enterprise computing*

| *Approach* | *Time Frame* | *Programming Model* | *Business Motivations* |
|---|---|---|---|
| Mainframe timesharing | 1960s–1980s | Procedural (COBOL) | Automated business |
| Client/server | 1980s–1990s | Database (SQL) and fat client (VB, Powerbuilder, etc.) | Computing power on the desktop |
| *N*-Tier/Web | 1990s–2000s | Object-oriented (Java, PHP, COM, etc.) | Internet/eBusiness |
| Service orientation | 2000s | Message-oriented (XML) | Business Agility |

## SUMMARY

Service orientation is a powerful concept and represents a business model that has been successful in a variety of industries (most notably the electric utility industry). Enterprises are in the process of evaluating service orientation and considering the potential that it holds for transforming the way business gets done and enabling an alignment between IT goals and business goals. Although the hype cycle is in full swing, there exist some tangible motivations and real-world value behind SOA. Throughout the remainder of this book, the subjects of service orientation and business alignment will be examined with a careful eye to identifying how a savvy business leader can determine when SOA makes sense and when it does not.

## SOA CASE STUDIES

A few examples will help you look at service oriented architecture (SOA) in proper context. Here, we will present two case studies. They will give you an idea of the type of business problems SOA is good at solving. We will also discuss the general solution approach.

Right off the bat, you will notice that these problems cannot be solved by software alone. You will need people, machines, and software all playing roles in a well-defined business process.

### Case Study A: Return Handling

Retail companies have been accepting sold goods back from their clients for a long time. This operation is generally called *return handling, goods inflow*, or *reverse inventory*. The case study presented here is based on the work done by de Koster et al.[4]

#### *General Background Information*

MO1 is a mail-order retail company. It sells electronic goods, such as television sets, home theater systems, CDs, DVDs and cables. MO1

runs an e-commerce web site where customers can place orders. MO1 also releases printed catalogues and accepts orders over the phone.

Mail-order companies experience a high rate of return. This is true for MO1. Customers return about 15% of the goods sold. About 20% of the warehouse space is dedicated toward returns handling.

### *Current Business Operation*

Products can be returned within 30 days of delivery. MO1 offers full satisfaction guarantee. If, for any reason, a customer is not happy with a product, all she has to do is call the customer support line. If the dissatisfaction is due to a perceived technical problem, customer support does its best to resolve them. If the customer confirms her decision to return, the customer service representative logs the reason for return and provides the customer with a return address.

All returns are sent to a warehouse. When a package arrives, a staff member locates the call center log for the order to find out why the product is being returned. What MO1 does with the returned products depends on the reason for return. Exhibit 1.4 summarizes the actions.

Exhibit 1.5 shows the current business process in a graphical form. A business process manager or analyst will typically model the business process this way using graphical notations.

### *The Problems*

Overall, MO1 needs to lower the cost of return handling and minimize errors. Specifically, the following problems exist in the current operation:

- When a returned package is received, it takes a staff member several minutes to locate the order details and the call center log. The staff attempts to locate the information by searching for the customer's name and address.
- Staff member has to manually enter the same data in several systems. These systems include call center, warehouse management

EXHIBIT **1.4** *List of reasons customers return items and the actions taken by the MO1 staff based on the reason for return*

| *Reason for Return* | *Action* |
|---|---|
| Product does not meet customer's need or expectation of quality. Product itself is not defective. | Product is touched up, repackaged, and returned back to shelf. The inventory on hand is incremented in the warehouse system. |
| Product is defective. | Return product to the manufacturer if the manufacturer accepts defective goods. Otherwise, discard product and book it in the accounting system as loss. |
| Product was damaged during shipping and handling. | File a claim with the shipping provider if the shipment was insured. Otherwise, discard product and book it in the accounting system as loss. |

system, manufacturer's web site, shipping carrier's web site, and the accounting system. This slows down the operation and introduces errors.

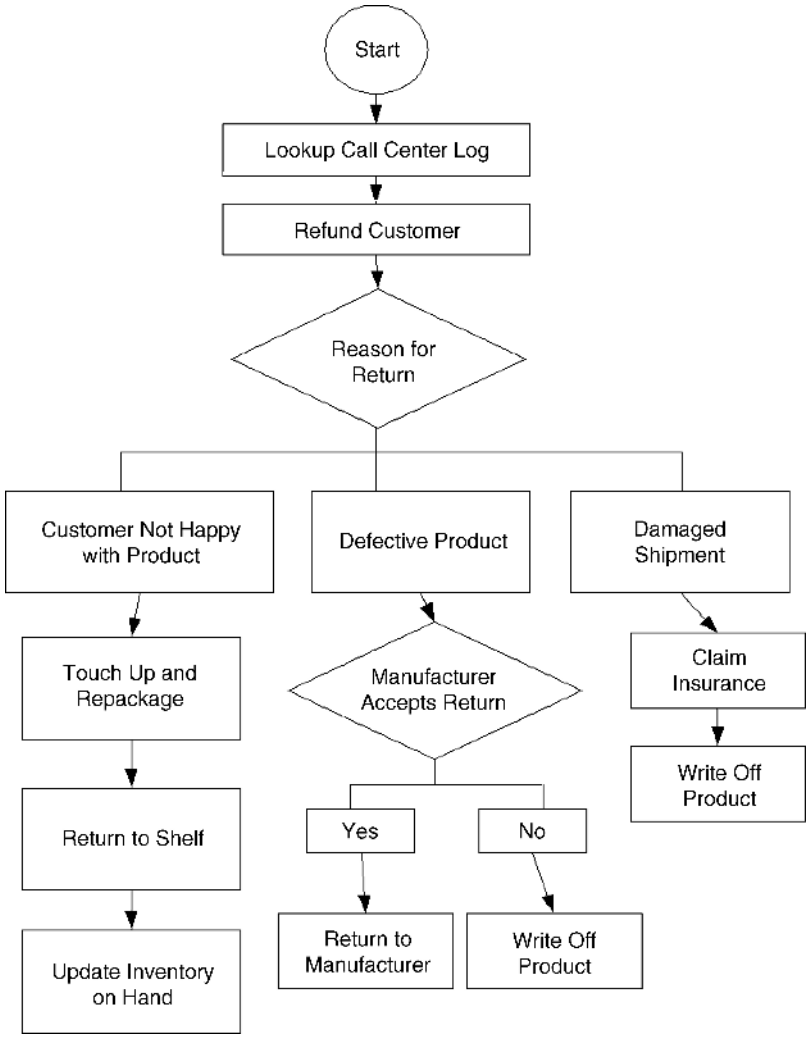
- Some of the defective items are sent back to the manufacturer. Staff member had made mistakes when preparing the package for shipment to the manufacturer. Shipping label has been printed with the address of the wrong manufacturer.
- Currently, staff members log into the accounting system and initiate refund to the customer. This has led to errors and in some cases malpractice.

### *Improvement Opportunities*

MO1 has identified several areas of improvement:

- It is generally believed that certain products have a higher rate of return. This may have to do with the way the product is represented in the web site or the printed catalogue. MO1 would like to know which products have a high return rate so that appropriate

EXHIBIT 1.5 *The as-is business process*




changes can be made to the description and photograph of the product.

- For some of the cheaper items, processing the return costs more than the item itself. In these cases, items could be simply scrapped.
- Return processing can be considerably sped up with a better-designed warehouse workflow and more efficient sorting process

for returned packages. The main focus area here will be integration between different systems, which will eliminate the need for duplicate data entry.

### *How Can SOA Help?*

Can SOA really help MO1 devise a solution?

SOA takes a much more complete view of the business problem than traditional approaches like object-oriented analysis and design (OOAD). SOA attempts to solve a problem by using a combination of employees, resources such as software applications, machines, factories, and business partners.

In a sense, SOA cannot be presented as merely a software development methodology. It is an overall problem-solving approach. It involves the entire business in building a solution.

Now, let us have a look at how one follows the SOA approach to solve these problems.

First, SOA requires that you take a close look at the business process. In case of MO1, the current return handling process is highly inefficient and prone to many errors. By performing *business process management* (BPM), we will be able to improve the process quite a bit. (BPM is discussed in more detail in Chapter 2, Business Process Management and SOA).

After MO1 redesigned the business process, the following improvements were made:

- In the packages shipped to the customers, include a label that should be used as the return address. This makes customers' lives easier. More important, the label contains a bar code identifying the order. When the returned package is received at the warehouse, the staff simply scans the bar code. There is no longer any need to search for the order using the customer's name and address.
- Once the bar code is scanned, the system should automatically pull up the call center log for the order. Staff member does not have to log into the call center application and search for the order.

- Staff member enters the reason for return in a database. This is later analyzed to fix problems such as errors in the web site.
- The workflow should automatically contact the accounting software and initiate refund.
- If the product is defective and the manufacturer accepts returns, the system should automatically send an invoice to the manufacturer and print the correct shipping label. All the staff members have to do is slap that label on the package and move it to the shipping section of the warehouse.
- If the package was damaged during shipping, the system should automatically file a claim with the shipping carrier.

In SOA, the business process itself executes as a software. It can automate many tasks, for example, pulling up the call center log and initiating a refund with the accounting system. The employees do not have to log into all kinds of different software applications and manually enter data.

Next, SOA requires that you formally identify the players in the process. They are called *service providers* or simply *services*. In the return handling process, the employees, the web site, the call center application, the accounting system, the manufacturer, and the shipping carrier are the services.

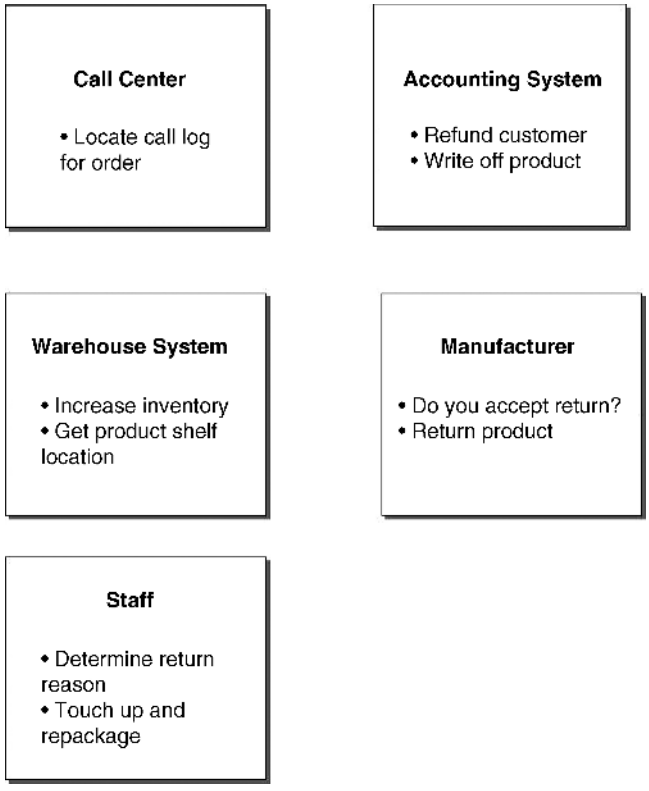
Each service performs a set of tasks. For example, the accounting system can be asked to refund the credit card used for an order. The shipping carrier can be asked to accept an insurance claim for damaged goods. The staff members accept returned packages, do touchups, and repackage. They also place items in good condition back on the shelf.

Exhibit 1.6 shows a list of services in the return handling process and the tasks they perform.

Once services are identified, their roles and responsibilities become very clear. SOA now requires you to look for ways to automate tasks. Task automation can significantly reduce error and cost.

If a service provider is a software, such as the accounting system, it is relatively easy to automate the tasks. Essentially a task is automated by developing the software for the service that performs that task.

EXHIBIT **1.6** *Example services identified from the return handling process. Each service shows a set of operations that it is capable of performing*




If the service provider is a human being, we need to look for ways to use software to automate the task. For example, we could have the system automatically launch the call center application and pull up the call log and order history before an employee begins processing a return. Not all human tasks can be automated. SOA and BPM recognize this reality and support human-based services as well.

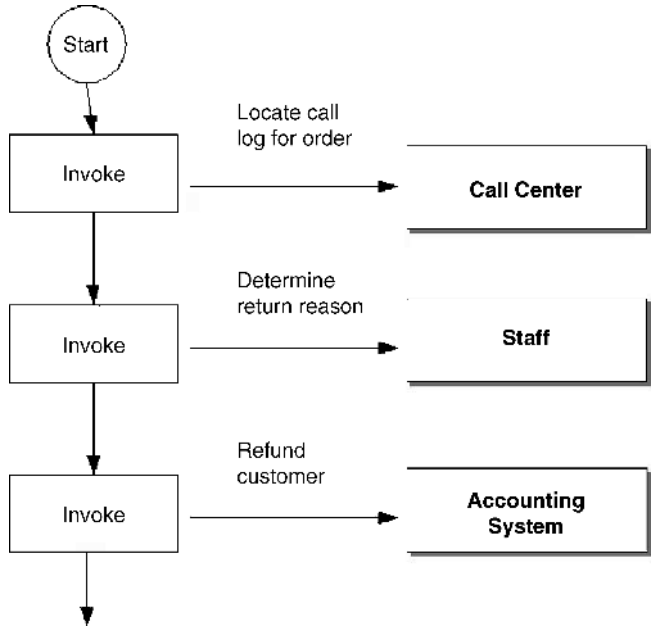
The shipping carrier and manufacturers are external organizations. They may or may not offer software services to automate the tasks. If they do, you need to consider using them instead of using their web sites, phone, or fax to ask them to do these tasks. For example, if the

shipping carrier offers a Web Service to file an insurance claim, we should consider using that.

Once all the services are implemented (either by using software or by assigning staff members the relevant tasks), we can develop orchestration. *Orchestration* is a software program that controls the sequence of tasks in a process. Each task is performed by a service. With orchestration, human beings no longer manage the flow of activities in a business. For example, if a product has been returned because it was damaged during shipping, the orchestration will automatically ask the shipping carrier service to file a claim. The orchestration knows the context, which is the original order placed by the customer. As a result, it can automatically fetch all the information required by the shipping carrier, such as the waybill number and the account number.

Exhibit 1.7 shows what an orchestration looks like. Usually, software developers take the business process model created by the process

EXHIBIT **1.7** *A snippet of the return handling orchestration*



managers or analysts and convert that into orchestration. Orchestration uses the Invoke activity to ask a service to perform a specific task.

MO1 wishes to get a better understanding about the return handling process. To aid that, we can define *key performance indicators* (KPIs) for the orchestration. For MO1 the relevant KPIs are:

- Percentage distribution of reason for return.
- The top products getting returned because they do not meet customers' expectations of quality or do not meet their needs.
- How long it takes end-to-end from the time a returned package is received to the time it is fully processed. This will give some idea about how well the process is working.

### *Solution Summary*

This completes our fictitious project where we follow the SOA approach to solve the problems of MO1. As you can see, SOA is not a technology, but a mindset for designing a solution. What follows is a summary of what we did:

- We used BPM to scrutinize the current business process. We found several ways to make it more error-proof and faster.
- We identified the roles and responsibilities of each service provider. Then we looked for ways to automate the tasks. We did that by developing services. Task automation can further speed up the operation and reduce error.
- We developed an orchestration. This will oversee the sequence of tasks. Orchestration has a twofold advantage. It helps with automation. It can also capture key statistical data known as KPIs, which help us get a better understanding about the business and improve its operations.

## Case Study B: Expense Approval

This case study shows how SOA can play a role in a small company or for a small project. HighTree is a fictitious company that provides

information technology (IT) training to Fortune 1000 companies. It employs several full-time and contracted teachers who travel to the training locations. These resources claim expenses incurred while they are on the road. HighTree has not been doing a very good job paying these claims on time or accurately. We will see how SOA can be used to improve the company's performance.

### *Current Business Operation*

Currently, a teacher files an expense by sending an e-mail to the salesperson. The salesperson approves or rejects the claim. About 99% of all claims are approved. If a claim is approved, the salesperson forwards the original claim e-mail to the accountant. If the teacher is an employee, the claim is paid in the next paycheck. Otherwise, a check is mailed to the contractor.

### *The Problems*

The business process for handling expense claims is simple and worked when HighTree was a small company. As the volume of claims has grown, a number of problems have started happening:

- It is common for busy salespeople to miss the claim e-mails from the teachers. These e-mails get buried amid hundreds of other e-mails that each salesperson gets.
- If a salesperson goes on vacation, approval gets delayed.
- Accountants made mistakes entering claim data in the accounting system. In some cases, income tax had been deducted from the payment. (Expenses are nontaxable benefits.)
- Before a teacher can get paid, she must send in the receipts. Correlating receipts received by mail to an expense item has caused major headaches for the accountants.
- Teachers have no clear idea whether or when a claim was paid. There have been cases where unpaid expenses have gone unnoticed.

### *Improvement Opportunities*

HighTree realizes that the expense claim process needs to be automated through a software-controlled workflow. The workflow should enter data into the accounting system. Manual data entry needs to be avoided as much as possible.

A new application needs to be developed that shows the current status of a claim. Teachers can use this software to view their claim history and make sure that they are getting paid on time.

### *How Can SOA Help?*

As we have seen in the previous case study, SOA encourages workflow and task automation. SOA will be a perfect fit for the problems facing HighTree.

First, we need to redesign the business process. Instead of sending an e-mail, a teacher will log into a web-based application and file a claim. We will call this the *Expense Management Application* (EMA). To file a claim, a teacher needs to:

- Select the teaching assignment for which the expenses were incurred.
- Enter the total claim amount.
- Create a Microsoft Excel file containing a list of all expense items.

Once a claim is submitted, EMA issues a claim number. When the teacher mails in the receipts, this claim number is shown on the envelope.

A salesperson logs into EMA and views a list of claims that she needs to approve. Here, the business process automatically routes the claim to the correct salesperson for the teaching assignment.

If the salesperson approves the claim, the process waits for the receipts to arrive. When receipts arrive, an accountant logs into EMA, enters the claim number from the envelope, and indicates that the receipts have been received. At this point, the workflow asks the accounting system to pay the teacher (either using check or payroll).

The payment is scheduled under nontaxable benefit so that no income tax is charged.

The workflow also informs the EMA about the payment. This helps EMA show the most up-to-date status of a claim to the teachers.

Once we have optimized the business process, we need to move to the service identification stage. The players in the process are:

- **The EMA application.** The workflow informs the application as the claim goes through different stages of its lifecycle. This allows a teacher to view the latest status of a claim.
- **The accounting application.** The workflow asks this application to pay a teacher.
- **Salesperson who approves or rejects claims.** The workflow waits for a salesperson to approve or reject a claim.
- **Accountant who receives receipts sent via mail.** The workflow waits for the accountant to enter a receipt before it asks the accounting system to initiate payment.

Note that a teacher initiates the process but does not play any role within the process.

Next, we move to the service implementation phase. Exhibit 1.8 shows you how each service will be implemented.

Finally, we build the orchestration. Orchestration will automate the business process. It will ask the services to perform their tasks. Some of the tasks are long running. Usually, human tasks are like that. The orchestration can wait for certain events to take place. For example, the orchestration waits for the receipts to arrive. In short, you should be able to use orchestration to implement most common workflow scenarios.

### *Service Maturity and Reusability*

This case study will help you understand how SOA fosters reusability. Let us have a look at the accounting system service. If we are implementing it for the first time, we will have to do the necessary work to implement the service. The same will happen if the service was already there but did not support the task of paying out nontaxable benefits.

EXHIBIT **1.8** *A list of services identified in the business process and how the services will be built*

| *Service* | *Implementation Notes* |
|---|---|
| EMA | Keep in mind that EMA is a web-based application. This needs to be developed from scratch. The development of this application is outside the scope of SOA and uses traditional software development techniques such as object-oriented analysis and design (OOAD).<br>However, EMA does have to implement a service. The tasks supported by this service are used by the workflow to keep the status of a claim up to date.<br>EMA will implement the service as a Web Service. |
| Accounting System | HighTree uses a popular third-party accounting software. An external application can ask the accounting system to perform all kinds of tasks by saving a file in proprietary data format in a particular directory. One of the tasks supported by the accounting system is to add nontaxable benefit to the next payroll of an employee. Another task supports paying a contractor by check.<br>In this case, we will implement the service using a file adapter. This is an example of how older software that does not offer a Web Service can still implement a service. |
| Salesperson | This will be a human-task service. Most SOA platform vendors allow you to develop such a service with little or no coding. The platform will also provide a web-based interface that a salesperson can use to view a list of task items (in our case, claims waiting for approval).<br>Most vendors also support escalation. For example, if a salesperson is on vacation, the system will automatically move the claim to her manager's pile. |
| Accountant | Same as above. |

We cannot avoid this upfront work, but with the SOA approach, the work can be minimal. SOA encourages us to use the functionality that is already built into the accounting software.

Over a period of time, we will keep adding more and more tasks to the service. This is called *service maturity*. As the service matures, it becomes more reusable. A new business process will most likely find that the service already supports a task that it needs.

### *Solution Summary*

The problem involved multiple applications and people. SOA is a great methodology to solve problems like this. The problem and solution matrix in Exhibit 1.9 paints the picture.

### Case Study Summary

These case studies show how the SOA approach is applied from the problem definition to the final solution. Examples are worth a thousand words. The examples here should help you understand these key points:

- SOA is a solution development approach that is ideal when the solution involves many software applications, people, machines, and business partners.

EXHIBIT **1.9** *List of problems in the case study and how they were resolved*

| *Problem* | *Solution* |
|---|---|
| E-mails missed by the salesperson | The EMA software tracks the claims. |
| Payment delayed because salesperson is on vacation | Human-task services support escalation. If a salesperson is on vacation, her manager will approve the claim. |
| Accountant enters wrong data | Payment information will now be automatically entered by the orchestration as nontaxable benefit. |
| Major problem correlating receipts with claim | Now the receipts will be marked with the claim ID. The orchestration will use this as a correlation identifier. When receipts are entered for a claim, it will process payment for that specific claim. |

- SOA takes a structured approach. First, the problems are identified. Then the business process is carefully optimized. Many of the problems are solved right at this point. Next, we identify the players or the services in the business process. Then we implement the services. Finally, we create an orchestration that automates the whole workflow.
- The line-of-business managers are actively involved in the early phases. They define the business processes and work on fixing its weaknesses. In the later phase, IT gets involved. They are responsible for implementing services and the orchestrations. SOA shines at bringing both sides to the same table. The process model and service specifications act as the common language.

## NOTES

1. The SOA layers depicted in this chapter are derived in part from IBM's component layering model for SOA. For more information, check out: www.ibm.com/developerworks/webservices/library/ws-soa-design1/.
2. Emphasis added is mine. To read the full analysis from Gartner regarding SOA's long-term viability, read: Core Research Note G00144445 by Yefim V. Natis, et al.
3. Enterprise computing timeline designed by Jason Bloomberg, Copyright 2004. Used with permission.
4. René B.M. de Koster, Marisa P. de Brito, and Majsa A. van de Vendel, ''How to Organise Return Handling: An Exploratory Study with Nine Retailer Warehouses,'' *Econometric Institute Report*, 2002, http://econpapers.repec.org/paper/dgreureir/2002264.htm.