Part 1 Overview

This first part of the book sets the context for the remainder of the book by explaining the principles of service orientation, as a business phenomenon enabled by developments in technology.

Chapter 1 provides a conceptual overview, defines the core terminology and introduces the pivotal idea of service-oriented architecture (SOA). Service orientation presents some massive cultural and technical challenges that cross three areas that have traditionally worked largely in isolation from one another: business process improvement, application development, and software operations. We provide a simple example of how the idea of a service can provide a unifying thread for drawing together these areas, along with some guidance for selling this approach to business management.

The remainder of part 1 maps out the foundation technologies required for practical application. Most of these technologies are developing out of previous well-established generations of technology. At the same time, they are based on standards that are complex as well as changing. Rather than attempting the impossible task of reviewing and documenting all of these technologies and standards we provide an essentially abstract primer designed for maximum resilience to the pace of change; a list of useful evolving Internet information sources is provided at the back of the book. For example, our emphasis is on the general idea of a *software service*, rather than specifically on a Web service, unless of course the context demands it.

Chapter 2 surveys the overall technology trends toward on-demand computing and sets Web services in context. We consider the execution management technologies and the gear shifts in service-level management (SLM) that are required in order to realize the promises of service orientation.

Chapter 3 parallels chapter 2, in that it considers the business process management (BPM) technologies and the associated gear shifts in business modeling that are required in order to effectively tackle the challenges of service orientation. In particular, we introduce the key concept of *service-oriented viewpoints*.

1 Basics of service orientation

1.1 The idea of service orientation

The phrase, “Easy to do business with” has become a major driver for most companies. The idea of service orientation is to provide customer value by contracting others to do what a company has to do just to get by, and by focusing the company’s own resources on what it does best. By subscribing to the *commodity functionality* provided by service providers who can perform it better, faster, and cheaper an organization can minimize its cost of market participation. This strategy releases energy for an organization to concentrate on its core competencies, thus bringing value to market through what the organization does best, thus making it easy to do business with. By taking a service-oriented approach, in an increasingly complex and competitive market, products can then be taken to market quickly and business processes conducted in agile response to change through multiple channels.

Such is the compelling attraction of service orientation. But “Wait a minute,” we hear top management say, “We might be serving the same customer, but we are a complex organization with multiple business units each with different targets, and supported by different technologies with different capabilities. We’ll let the faster-moving, more profitable business units continue to do their own thing and just outsource the rest without worrying too much about the provider. That way we’ll be service-oriented but we can’t afford an integrated approach.”

However there is no free lunch – current practices are simply not going to get them there. The main business driver of service orientation is *agility*: the speed, cost-effectiveness, accuracy, and flexibility required for organizations to prosper. Achieving agility requires a service-oriented architecture (SOA) that aligns technology with business goals, allowing the company to move in new and exciting ways that open up new markets. Very briefly,[1] an SOA refers to the software structures and policies that are required to enable the business phenomenon of service orientation. Managing agility requires appropriate processes that connect consumers and providers of services in a

[1] A great deal more detail is provided as the book unfolds. SOA is introduced further in this chapter and is the central topic covered in part 3.

cohesive fashion. Above all, service orientation requires effective execution strategies for dealing with inefficient ingrained business processes and the legacy software that holds them together. In this chapter, we introduce the concepts that are necessary to understand the change in mind-set mandated by service orientation.

1.1.1 The economic imperative

Increasingly, many of our business leaders see IT as a cost that must be trimmed to the bone. As a result of the ongoing quest to cut costs, more and more organizations are outsourcing large slices of both software development and business operation. This “economic imperative” is most strongly embodied in an article in the May 2003 edition of the *Harvard Business Review* by Nicholas G. Carr (Carr, 2003), who contends that IT has ceased to provide competitive advantage. He compares the growth of IT through the late twentieth century to the global growth of rail track in the mid-to-late nineteenth century and with the expansion of electric utility generating capacity in the U S in the early part of the twentieth century. Periods of massive growth in these industries followed huge investment but subsequently resulted in falling prices and in commoditization. Carr argues that IT has similarly become a commodity today, and that companies should therefore look to spend much less on it.

You may well, like me, see this argument as based on a flawed view of IT as nothing more than a set of bytes that is part of market participation cost – for example, you simply have to use human resources (HR) software to pay your employees; Smith and Fingar (2003a) provide a particularly good counter argument. At the same time, it is important to realize that the overgeneralization implicit in this kind of argument is alive and flourishing today in the minds of many of our business leaders. It is important therefore that CIOs and senior architects equip themselves with clear concepts and good business arguments as they seek to convince their business leaders of the need for better software architectures.

One of the most important tasks of CIOs and senior architects is to articulate to the business leaders why Carr’s argument is a danger to the organization’s health. In particular, it is important to explain that while it is true that some aspects of IT are indeed commoditized that does not mean that companies no longer have any IT applications that produce value for them that their competitors cannot easily copy.

At the same time, there is an important measure of truth in Carr’s argument that we would be foolish to ignore. There is an economic imperative that is forcing many large organizations to become smaller and more specialized as they focus on what they do best and on what differentiates them from the competition. In turn, this involves outsourcing many of the support functions that have traditionally been managed in-house to third parties who can provide these services more cost-effectively and at higher quality through their own focused competencies. For example, a whole business process may be outsourced to others who are better equipped to deal with it efficiently. Payroll is the classic example. Another is the insurance company that chooses to outsource its claims function. In essence, these "treadmill" services simply represent the cost of market participation. Utility work such as call centers, billing, and claims processing is – courtesy of the Internet – following manufacturing jobs by transferring to places such as China, India, and the Philippines and other low-wage countries.

These business trends in service orientation are paralleled at the IT level. For example, packaged software, from small components to large-grained enterprise resource planning (ERP) solutions, provides a long-standing example of automated treadmill services. There is also a groundswell of legacy software, from COBOL systems to back-end databases that perform much of the treadmill work. In addition, the running of this software may be outsourced to an application service provider (ASP). The Internet and now the emergence of utility computing (Hughes, Bader, and Corrigan, 2004), which we discuss shortly, is accelerating this trend of focusing on one's core competencies and letting providers do the rest.

The problem that many organizations face today is that because their current software is not organized in a service-oriented way they have lost control of their software. And, in some cases, where economic pressures reach boiling point, it is all too easy to succumb to the "IT Doesn't Matter" line of argument and outsource all of IT: "Act in haste, repent at leisure" is a maxim that comes to mind. These organizations do not have a clear picture that allows them to make the right decisions on service provision. The SOA is designed to provide that picture.

1.1.2 The competitive imperative

If becoming "leaner and meaner" is one side of the service-oriented coin, the need to compete and to attract and keep customers is the other. Customers are growing more service-oriented in that they are demanding much smoother and better experiences from the companies that they choose to do business with. Forward-thinking companies are applying service orientation in using technology innovatively to support the customer experience.

For example, it is one thing to integrate existing legacy systems so that their services are exposed consistently across different channels. However, what we are now seeing is the emergence of new *multi-channel business processes*, in which a business process is supported by a variety of channels along its route from inception to closure. For example, like many others, I now purchase my air tickets by credit card online via my laptop and then use my credit card to complete the transaction at a self-service airport kiosk, which dispenses my tickets and itinerary without the need for me to suffer a long wait at the check-in queue.

Amazon.com has evolved into a multi-channel retailer, interacting with customers using Web portals, call centers, email contact centers, and catalogs. Through kiosks placed strategically in its various partners' stores, Amazon.com even has a significant presence in the bricks-and-mortar channel.

These kinds of approaches require significant investment in the SOA and the associated technologies required to manage and run it. As a business process changes, so must the services supporting it. Moreover, customers from any channel value *visibility*. For example, they want to know the status of their order. The Internet has further shaped customer expectations by enabling visibility into the process. However it is the quality of the underlying SOA that determines the success of the process.

In addition, those companies that become the most successful in their channels open up further opportunities for themselves. For example, a growing number of organizations including Toys 'R' Us, Circuit City, Target, Office Depot, and Nordstrom, are outsourcing their online channel interaction to Amazon.com. In a relatively short period, Amazon.com has built a lucrative new outsourcing business selling goods on behalf of other merchants. Amazon.com collects commissions on those third-party transactions with neither the risks nor the costs of owning inventory. Amazon's example presents compelling evidence that a service-oriented approach can deliver the strategic advantage that simply breaks apart conventional business models.

While Amazon.com is perhaps an extreme example, it is indicative of the overall change that is taking place in the way businesses are organized. In the 1980s, businesses were organized along strict departmental lines. In the 1990s, the emphasis shifted toward end-to-end processes, or value chains, in an effort to streamline and optimize, but still very much within the traditional boundaries of the business. The "noughties" are witnessing a further evolution in the form of federated processes[2] fueled by the emergence of the economic and competitive imperatives. This is the world of service orientation in which traditional boundaries quickly disappear and in which new "virtual" boundaries can emerge in response to new opportunities and changes such as mergers and acquisitions (M&As).

The economic factors that underlie the movement toward service orientation are extremely compelling in the long run. In most industries, specialization occurs as the industry matures. Those who focus on providing a particular kind of service and learn to do it best end up doing it on behalf of others. By acquiring commodity services at the lowest cost, an organization can focus much more strongly on its core competencies that add most business value. Equally there are competitive factors that underlie the movement toward service orientation in the form of multi-channel processes and the agility to compete in different channels. A company needs the ability to take its core competencies to market in new and innovative ways that beat the competition.

The move toward a federated business model involves dealing with the often conflicting drivers of economic stability and competition. On the one hand, companies

[2] As we examine in more detail in 1.3.1, a federated process allows largely independent parts to act with the unity of a whole, toward a common purpose, such that the whole is more than the sum of the parts. In contrast, a tightly integrated process, involves highly dependent parts.

require constructive strategies for dealing with the trend toward commoditization of software services that is fueled by the economic imperative. On the other hand, they need techniques for applying software services to improve the experience of customers and to extend business in competitive fashion.

Organizations are improving business processes wherever possible to drive productivity higher and compete more effectively. However, even if isolated parts of business processes are performing well, too often the business process as a whole is not. That is because many processes – with customers, suppliers, providers, partners, and employees – remain largely disjointed, papered together with a myriad of information flows: telephone calls, faxes, spreadsheets, and FedEx packages. Connecting and improving these fragmented processes has taken on a new urgency due to the speed and change of business today toward service orientation.

1.1.3 The legacy jungle

Of course, these changes in business process do not happen in a vacuum! Many organizations are faced with hybrid software and hardware environments that have evolved over a number of years. Most of these organizations are not early adopters: there is a need to minimize risk and maximize business return from existing software assets, both internally developed and externally acquired. In many situations, these software systems are acquired from and outsourced to a wide variety of vendors. These external systems need to be interfaced with many internally developed systems. The result is usually a complex patchwork of applications, a "legacy jungle" if you will, that is costly to maintain, inflexible to change, and causes unacceptable development times.

At the same time, these systems have stood the test of time: they "do a job." They may not do it in the most adaptable and efficient manner. They may be a maintenance nightmare. Nevertheless they are tried and tested. There is a comfort factor embodied in the slogan "Better the devil you know." The fact is that "legacy" is not a bad term, despite the fact that it is often read that way in the context of software. A legacy represents something bequeathed by a predecessor that has a value not commonly apparent. There is an onus on the inheritor to do some work to realize the value. As well as facilitating the trend toward federated processes a service-oriented approach must also support low-risk migration and integration of the legacy software portfolio to achieve ease of maintenance, flexibility, and responsive solution delivery.

1.1.4 The need for balance

Cutting a path out of the legacy jungle is a pressing challenge for most IT departments especially in large organizations. Enterprise application integration (EAI) technology can provide some help, but this is limited by the proprietary nature of the products. The introduction of *portals* that provide customers or users with a consolidated view

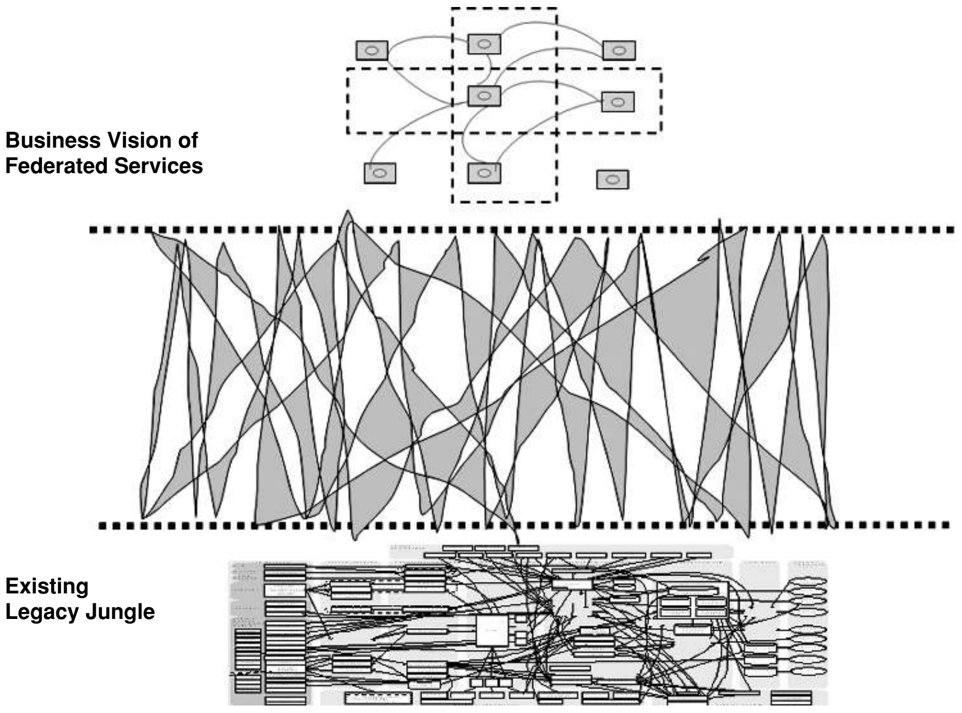

Figure 1.1 Severe churn across the business–IT gap

of business functionality can provide some relief. At the same time, such “integration on the glass” must be seen for what it is: a short-term measure. The danger with such approaches is that a whole new generation of legacy systems is born as portals spring up on a local departmental basis with little heed to adaptability, consistency, or reuse.

The vision of federated services (discussed in 1.3.1) is one in which a firm can respond with agility to changes in business scope, as illustrated by the dashed boundaries in figure 1.1. However, the gap between the vision of where the business wants to go and the current state of IT only widens and grows more difficult to address with each layer of technology that is applied to the problem. “Getting anything done round here is just such a hassle,” complain the business users. A single software change request can have multiple knock-on effects resulting in severe “churn” across the business–IT gap, as illustrated in figure 1.1.

Only by aligning its supporting software with its business processes in such a way that software can be reused across those processes and then replaced when things change is it possible to reduce churn and provide long-term balance. Design of the SOA is the discipline that organizations must master if they are to achieve this balance, and if they are to win and maintain control of their software portfolio and ensure that it aligns with business needs.

1.2 Some definitions

At this point, we need to take a brief step back and firm up on some definitions.

1.2.1 Services

A *service* is offered by a provider to a consumer through its *interface*. An interface describes the contract between the provider and the consumer. In other words, it should specify what the provider is obliged to do on behalf of the consumer and what responsibilities the consumer agrees to in using the interface. Everyday life is full of examples of service interfaces. Sometimes these are written down in detail, as in the case of buying a TV and receiving a product booklet and written guarantee. At other times the interface is assumed, as in the case of a car wash (although it is usual to see a brief description of what types of wash are on offer as well as instructions describing what the driver must do to use the car wash).

A **service** is functionality that must be specified in the business context and in terms of the contracts[3] between the provider of that functionality and its consumers. Implementation details should not be revealed. The implementation of the service does not have to be automated – it could consist of purely human activity.

The concept of a service is akin to a reusable chunk of a business process that can be mixed and matched with other services.

1.2.2 Business processes

The traditional view of a business process is as follows.

A **business process** is a set of activities that is initiated by an event, transforms information or materials, and produces an output. These sets of activities are either value chains that produce outputs valued by customers or infrastructure processes that produce outputs that are valued by other processes.

[3] Whereas functionality is traditionally expressed in terms of procedural steps (if A then do B, then do C until $X = Y$) , the contracts of a service are expressed nonprocedurally (given input A with $K = L$ then output B with $M = N$ is produced).

Perhaps the most significant aspect of the role of services is that they start to reshape this view of business processes, taking us toward a more *federated business model.* A business process is usefully pictured as being composed of re-configurable services.

1.2.3 Software services

A **software service** is a type of service that is implemented by software and that offers one or more operations (or software functions).

In this sense, a software service becomes a commodity bought, sold, and delivered in a similar manner to any other kind of service – for example, electricity or telecommunications. The consumer of a service is not concerned with implementation detail. The implementation of the service can vary from one supplier to another while still delivering the same service through its interface. At the same time the consuming software, invoking the service, can be implemented using any technology we choose providing it calls the services using the right interface. This situation is illustrated in figure 1.2. The same service, Transfer Funds, is provided by two alternative suppliers, each of which uses different implementations (COBOL on a mainframe, and C++ on a Unix platform). This service may be invoked via three different channels, cell phone, ATM, and PC. The implementation is transparent to the users of the service.

The potential for reuse of the same software service in different contexts is readily apparent here. This is especially important when a business wants to move a software service into a new sales channel. Equally, one implementation technology is replaceable with another. The possibility of upgrading to better technologies without disruption is hugely attractive. And from a utility computing perspective, discussed in chapter 2, the possibility of being able to switch implementations according to various criteria (for example cost, level of quality, or user demand), while retaining the same functionality is again hugely attractive.

1.2.4 Web services technology

Web services technology is a set of XML-based industry standards and specifications that specify a communication protocol,[4] a definition language,[5] and a publish–subscribe registry.[6]

[4] Simple Object Access Protocol (SOAP). [5] Web Services Description Language (WSDL).
[6] Universal Description, Discovery, and Integration (UDDI).

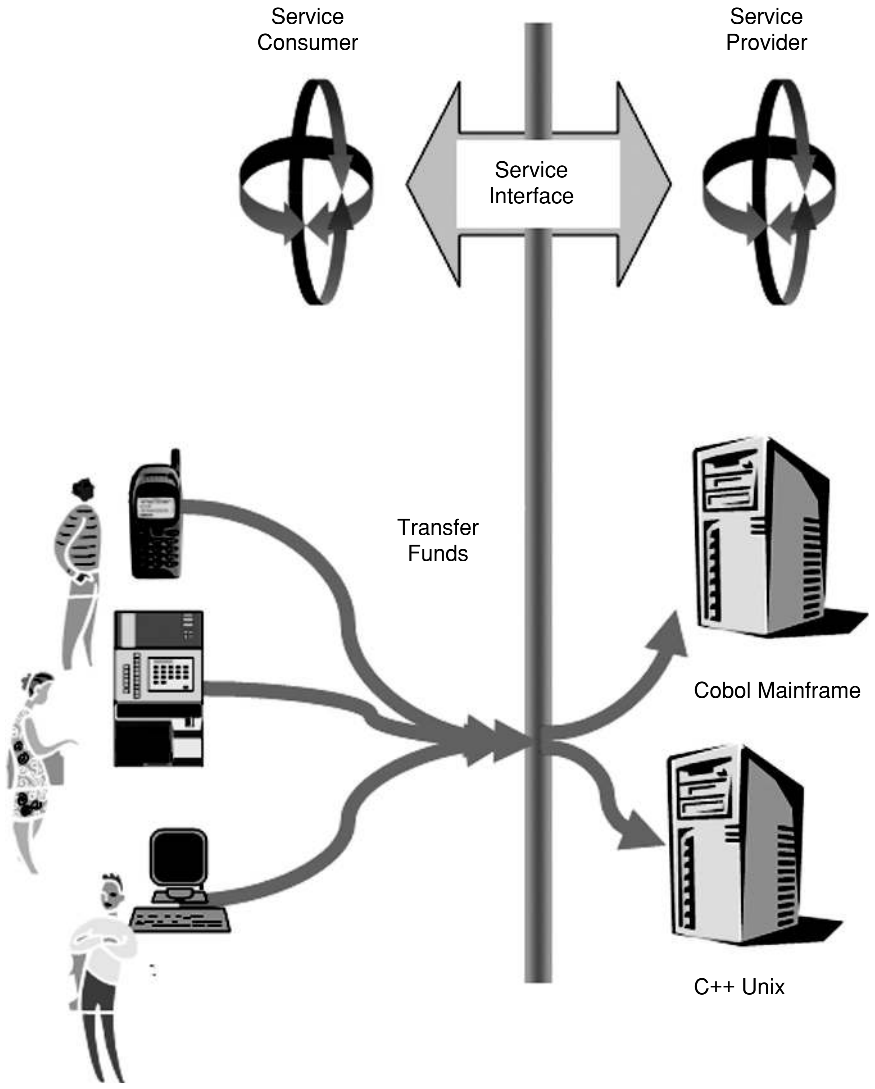

Figure 1.2 Multiple consumers and suppliers of the same service

These are the core features of Web services, though there is a proliferation of associated emerging standards required for industrial-strength application. Web services provide a particularly good means of offering software services, in that they provide the enabling technology standards for languages, protocols, and registries. However, it is perfectly conceivable to use other (or alternative) enabling technology standards – who knows what is around the corner in the world of emerging technology standards? There is nothing sacrosanct about Web services. Therefore in this book we use the general term "software service" unless the context demands otherwise. At the same time, because