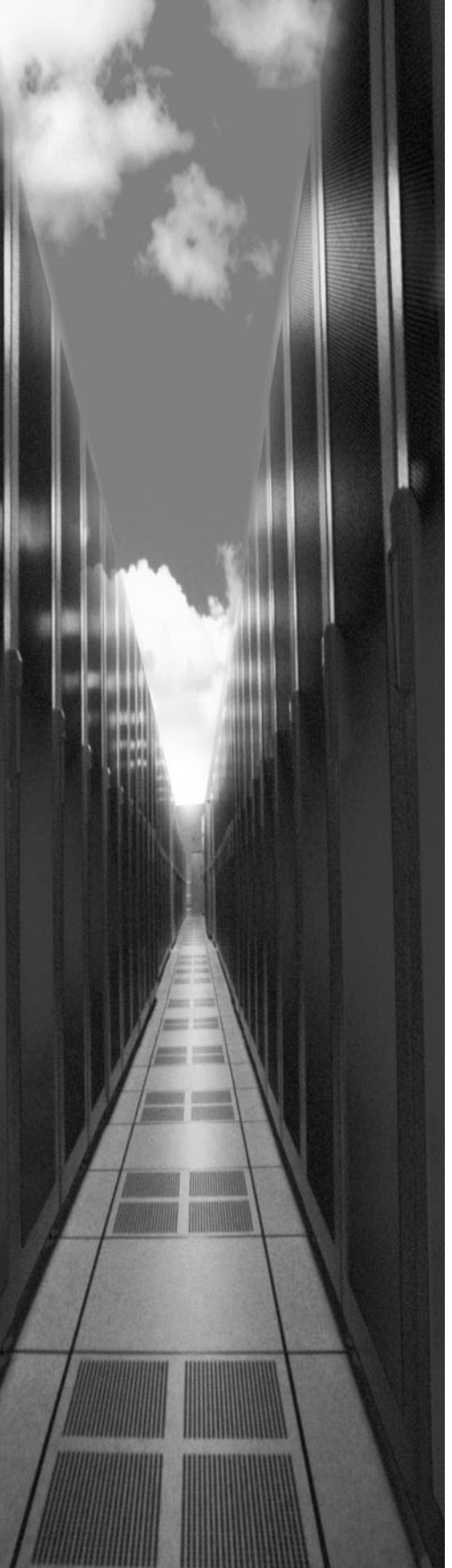

CHAPTER 1

Where We Are, How We Got Here, and How to Fix It

There are painters who transform the sun to a yellow spot, but there are others who with the help of their art and their intelligence, transform a yellow spot into the sun.

—Pablo Picasso (1881–1973)

It is Thursday morning, you are the CEO of a large, publicly traded company, and you just called your executives into the conference room for the exciting news: the board of directors has approved the acquisition of a key competitor, and you are looking for a call-to-action to get everyone planning for the next steps.

You talk to the sales executives about the integration of both sales forces within three months, and they are excited about the new prospects. You talk to the human resources director, who is ready to address the changes HR must make within two months. You speak to the buildings and maintenance director, who can have everyone moved who needs to be moved within three months. Your heart is filled with pride.

However, when you ask the CIO about changing the core business processes to drive the combined companies, the response is much less enthusiastic. "I'm not sure we can change our IT architecture to accommodate the changes in less than 18 months, and I'm not even sure if that's possible," says the CIO. "We simply don't have the ability or the capacity to integrate these systems. We'll need new systems, a bigger data center. . . ." You get the idea.

As the CEO, you are nonplussed. While the other departments are able to accommodate the business opportunity in fewer than four months, IT needs almost two years?

In essence, IT has become the single-most visible point of latency when a business needs to change. Thus, the ability to change is limited by IT. In this case, the merger is not economically feasible, and the executive team is left scratching their heads. They thought IT was about new ways to automate the business and had no idea how slow the IT folks are to react to change.

However, it does not have to be this way. The survival of many businesses will depend on a fundamental change in the way we think about and create our IT infrastructure going forward—that is, if you are willing to admit where you are and are willing to change. There is much work to be done, and reading this book is a great first step.

How Things Got Off Track

IT issues are best understood by understanding its general history over the last 30 years as well as your company's specific IT history. History tells you why things are they way they are. Examining your company's IT history is almost like participating in a 12-step program: you admit you have a problem and are willing to look at how you got here.

It is also important that you check your ego at the door. IT folk typically do not like to talk about mistakes made in the past. Indeed, many will defend until the day they die all IT-related decisions that were made in the past. But the point of examining the IT history is not about placing blame—it is about recognizing what you are currently dealing with and discovering ways to fix it. If you cannot open your eyes and mind to the existing problems, then reading the rest of this book will do you very little good.

If there is one issue that comes to mind each and every time we look at IT's past mistakes, it is *managing-by-magazine.* In essence, those charged with building and managing IT systems often did not look at what was best for the business but looked instead at what was most popular at the time or at what was being promoted in the popular computer journals as the technology "required" to solve all problems.

Another issue is *managing-by-inertia*, or failing to do anything just because it is new and unknown. This problem is opposite to *managing-by-*

magazine: Instead of doing something just because it is popular, we simply sit on our existing IT architecture. Typically, this lack of action is rooted in the fear of change and the risks associated with it.

We had the structured computing revolution, which became the object-oriented computing revolution, which became distributed objects, which became component development, which became enterprise resource planning, which became customer relations management, which became service orientation—you get the idea. Of course, I am missing a bunch of other technologies that we "had to have," including data warehousing, business intelligence, business process management—the list goes on and on.

Not that these technologies were bad things; most were not. However, they had the effect of distracting those in IT from the core problems of their business and focusing their attention more on the productized technology than on the needs and requirements of the business. The distraction was easy because analyzing and documenting business requirements was not as fun as experimenting with new technologies and was not a résumé-enhancing experience.

This focus more on the solution than on the problem caused a layering effect within the enterprise architectures. In essence, the architectures grew more complex and cumbersome because the popular products of the day were being dragged into the data center and became another layer of complexity that both increased costs and made the enterprise architecture much too fragile, tightly coupled, and difficult to change.

Today we have IT infrastructures and enterprise architectures that are just too costly to maintain and difficult to impossible to change. As business needs change, including upturns and downturns in the economy, IT is having an increasingly harder time adjusting to meet the needs of business. As in our example at the beginning of this chapter, CEOs are finding that IT is typically the latency within the business that causes delays and cost overruns, and IT does not add value to the enterprise as it once did. Remember when IT was the solution and not the problem?

IT departments were more productive when they were coding applications in COBOL on mainframes because it required them to be lean and cautious with their use of resources. Today, we have almost too much technology and too many options. We gave IT enough rope to hang themselves, or at least to get their IT architectures in a state that makes them much less valuable to the business.

SOA to the Rescue?

While there are many attempts to fix the badly broken IT architectures within our enterprises, most "solutions" just put another technology layer on top of the existing technologies in hopes that the technology will somehow fix the issues. As you may have guessed, it just makes things more complex. Few enterprises were willing to take the risk and address the core issues.

Service-oriented architecture, or SOA, is really about fixing existing architectures by addressing most of the major systems as services and abstracting those services into a single domain where they are formed into solutions. Simple in concept—and really nothing new—SOA is our best approach to fixing the broken architectures. With the wide use of standards such as Web Services, SOA is being promoted as the best way to bring architectural agility to your enterprise—that is, if you do SOA correctly. There is no magic bullet here.

SOA is a valid approach to solve many of the architectural problems that enterprises face today. However, those who implement SOA typically look at SOA as something you buy, not something you do. Thus, many SOA projects are again about purchasing some technology that is sold as "SOA-in-a-box," which turns out to be in-a-box but not SOA, and thus only adds to the problems.

SOA, as the *A* implies, is architecture. And thus it is the orderly arrangement of systems that best serve the service needs of the business. Taken in its literal context, enterprise IT can succeed with SOA. However, most often it does not succeed, and much of that failure occurs because SOA implementers view SOA as something other than architecture, and most often those implementers are not architects.

SOA is a valid architectural pattern and one that is leveraged throughout this book, but you need to look at SOA as a journey, not a project, and clearly not a product. At the same time, you need to break SOA down into small, incremental successes that also move the enterprise toward the core value proposition of SOA, which becomes even more powerful when leveraged with emerging concepts such as cloud computing, the other focus of this book.

We can call this "small SOA" and "big SOA." Big SOA encompasses the larger strategic objectives of SOA: simultaneously moving all the enterprise IT assets to something much more agile and easy to change. An example would be breaking down all relevant enterprise systems to a functional primitive, building them up again as services, and adding a process configu-

ration layer to form solutions. Considering that such an enterprise typically means hundreds and sometimes thousands of systems, this project could take many years.

Small SOA is just an instance of a big SOA. Small SOA is still SOA but has well-defined objectives, a time box, and a core return on investment that must be met. The lesson here is to leverage small SOA to get to big SOA. For instance, you could build a partner portal using SOA approaches that you can stand up in 6 months, with a return on investment of only three months—clear benefit and a small and doable project that occurs within a year's time.

While small SOA seems to be working, big SOA has largely been abandoned as too complex and too expensive to pursue. The reality is that you need both, but you need to know how to leverage both. Hold on to that thought for now. The topic of SOA is revisited many times in this book.

What the Heck Is SOA, and Why Should I Care?

First, let's put forth my definition of SOA so we have a foundation to work from.

> An SOA is a strategic framework of technology that allows all interested systems, inside and outside of an organization, to expose and access well-defined services, and information bound to those services, that may be further abstracted to process layers and composite applications for solution development. In essence, SOA adds the agility aspect to architecture, allowing us to deal with system changes using a configuration layer rather than constantly having to redevelop these systems.
>
> The primary benefits of an SOA include

1. Reuse of services and behaviors, or the ability to leverage application behavior from application to application without a significant amount of recoding or integration. In other words, SOA enables use of the same application functionality (behavior) over and over again without having to port the code, leveraging remote application behavior as if it existed locally.
2. Agility, or the ability to change business processes on top of existing services and information flows, quickly and as needed, to support a changing business.

3. Monitoring, or the ability to monitor points of information and points of service, in real time, to determine the well-being of an enterprise or trading community. Moreover, SOA provides the ability to change and adjust processes for the benefit of the organization in real time.
4. Extended reach, or the ability to expose certain enterprise processes to other external entities for the purpose of interenterprise collaboration or shared processes. SOA can be used as a key technology-enabling approach to leverage cloud computing (described later in this chapter).

The notion of an SOA is not at all new. Attempts to share common processes, information, and services have a long history, one that began more than 10 years ago with multitier client/server—a set of shared services on a common server that provided the enterprise with the infrastructure for reuse and now provides for integration—and the distributed object movement. *Reusability* is a valuable objective. In the case of SOA, it is reuse of services and information bound to those services (see Figure 1.1). A common set of

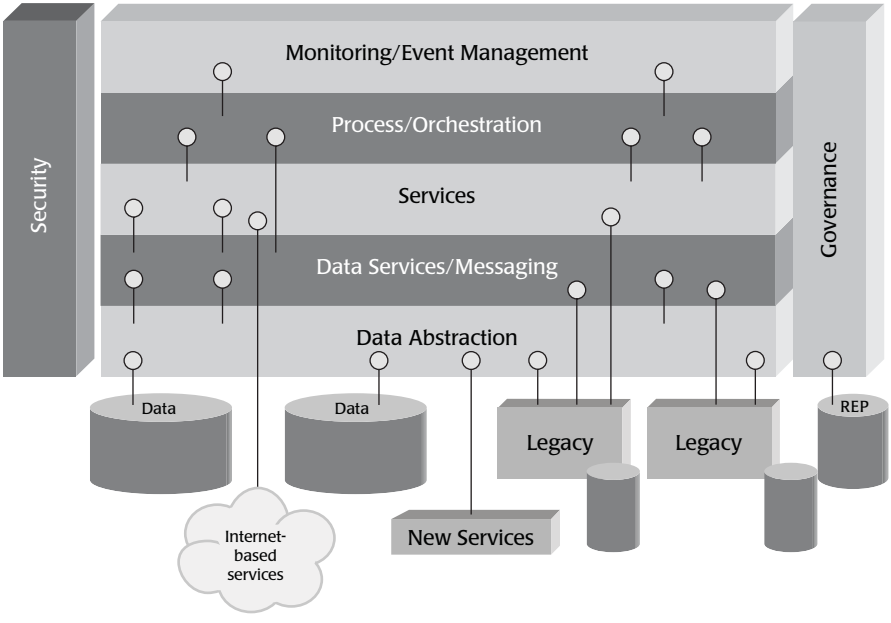

Figure 1.1 SOA metamodel provides a good way to see how SOA leverages a process/orchestration layer to change major business processes without driving changes to all systems. This is a loosely coupled architecture.

services among enterprise applications invites reusability and significantly reduces the need for redundant application services.

What is unique about SOA is that it is as much a strategy as a set of technologies, and it is really more of a journey than a destination.

SOA Meets Cloud Computing

So, what does SOA have to do with cloud computing, and why did we write a book about it? Cloud computing is any IT resource, including storage, database, application development, application services, and so on, that exists outside of the firewall that may be leveraged by enterprise IT over the Internet. The core idea behind cloud computing is that it is much cheaper to leverage these resources as services, paying as you go and as you need them, than it is to buy more hardware and software for the data center. There are other advantages as well.

Cloud computing allows you to expand and contract your costs in direct proportion to your needs. Moreover, it shifts some of the risk around expanding your IT resources from the enterprise to the cloud computing provider. We cover the business benefits of cloud computing in Chapter 4, "Making the Business Case for Clouds." Also, cloud computing abstracts those using the cloud computing–delivered IT resource from the management of those resources.

The relationship between cloud computing and SOA is that cloud computing provides IT resources you can leverage on demand, including resources that host data, services, and processes. Thus, you have the ability to extend your SOA outside of the enterprise firewall to cloud computing providers, seeking the benefits already described. We describe this process as "SOA using cloud computing," and it is the objective of this book to show you how it is done.

SOA is important to cloud computing for a few key reasons:

- It is a good approach to architecture that deals with the proper formation of the information systems using mechanisms that make them work and play well together, inside and outside of the enterprise.
- In order to take advantage of cloud computing, you need interfaces and architectures that can reach out and touch cloud computing resources. While many believe they can simply create quick and dirty links between core enterprise information systems and cloud computing resources, the

fact is that you really need an architecture inside of the enterprise, such as SOA, to make the most of cloud computing. That is the theme of this book.

- You need some sort of architectural discipline with guiding principles to document and organize your architecture. Most have ignored this need over the past several years to focus on ad hoc hype-drive stuff. We must get back to leveraging the best solution for the problem, and SOA is a good approach for doing that if you follow the steps.

For our purposes, we know that cloud computing is the ability to provide IT resources over the Internet. These resources are typically provided on a subscription basis that can be expanded or contracted as needed. This includes storage services, database services, information services, testing services, security services, platform services—pretty much anything you can find in the data center today can be found on the Internet and delivered as a service.

If you think you have seen this movie before, you are right. Cloud computing is based on the time-sharing model we leveraged years ago before we could afford our own computers. The idea is to share computing power among many companies and people, thereby reducing the cost of that computing power to those who leverage it. It was a pretty simple idea at the time. The value of time share and the core value of cloud computing are pretty much the same, only the resources these days are much better and more cost effective. Moreover, you can mix and match them to form solutions, which was not possible with the traditional time-sharing model.

There is nothing to fear from cloud computing. Indeed, it should be comforting to leverage resources that you do not have to maintain. Moreover, the sharing model has been around for years—we just call it something new: cloud computing. There are also some new offerings in this space that we discuss next.

The opportunity to learn how to leverage cloud computing—in the context of well-known architectural approaches such as SOA—is a way to get your enterprises leveraging a more efficient and effective IT infrastructure. However, cloud computing is not a cure-all or something that you attach to your systems and hope for the best. You have to do some planning to leverage cloud computing resources in the right way. In essence, that is what this book is about.

Defining Cloud Computing

While cloud computing is widely defined, we need a standard definition for the purposes of this book. The National Institute of Standards and Technology (NIST), Information Technology Laboratory, provides the most comprehensive definition of cloud computing thus far offered.

Cloud computing is a pay-per-use model for enabling available, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and comprises five *key characteristics:*

- *On-demand self-service.* A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed without requiring human interaction with each service's provider.
- *Ubiquitous network access.* Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and PDAs).
- *Location-independent resource pooling.* The provider's computing resources are pooled to serve all consumers using a multitenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. The customer generally has no control over or knowledge of the exact location of the provided resources. Examples of resources include storage, processing, memory, network bandwidth, and virtual machines.
- *Rapid elasticity.* Capabilities can be rapidly and elastically provisioned to quickly scale up, and rapidly released to quickly scale down. To the consumer, the capabilities available for rent often appear to be infinite and can be purchased in any quantity at any time.
- *Pay per use.* Capabilities are charged using a metered, fee-for-service, or advertising-based billing model to promote optimization of resource use. Examples are measuring the storage, bandwidth, and computing resources consumed and charging for the number of active user accounts per month. Clouds within an organization accrue cost among business units and may or may not use actual currency.

Note that cloud software takes full advantage of the cloud paradigm by being service oriented with a focus on statelessness, low coupling, modularity, and semantic interoperability.[1] However, all cloud computing approaches are not the same, and several deployment models, while different, are still considered cloud computing:

- *Private cloud.* The cloud infrastructure is owned or leased by a single organization and is operated solely for that organization.
- *Community cloud.* The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations).
- *Public cloud.* The cloud infrastructure is owned by an organization selling cloud services to the general public or to a large industry group.
- *Hybrid cloud.* The cloud infrastructure is a composition of two or more clouds (internal, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting).

Each deployment model instance has one of two types: internal or external. Internal clouds reside within an organization's network security perimeter, and external clouds reside outside the same perimeter. For the purposes of this book, we focus primarily on public cloud computing, or the use of a public cloud provider or providers to host portions of our SOA. Many businesses will find that private clouds are a better solution for their situation, leveraging the benefits of cloud computing but within their firewall. Or, they may choose to leverage a mixture of public and private clouds, or a hybrid cloud. Finally, some may create semiprivate or community clouds, which are public clouds leveraged only by a closed group of companies or government agencies.

The message is that it is all cloud computing and all of these architectural options are available to you. The steps highlighted in this book are applicable to private, community, and hybrid cloud computing[2] as well as to public cloud computing. We cover private clouds in a bit more detail in Chapter 12, "Moving Onward."

1. http://csrc.nist.gov/groups/SNS/cloud-computing/index.html
2. Ibid.

The Components of Cloud Computing

As cloud computing emerges, there is a lot of discussion about how to describe it as a computing model. Maturity models have been published and debated, and providers clearly have a model for their own products.

In attempting to better describe cloud computing, we came up with a "stack" of sorts, which logically considers each component of cloud computing and how the components interact. While clearly this model could be much more complex, it does not need to be. This stack explanation is a model for defining and refining the concept of cloud computing (see Figure 1.2).

While many in the industry can debate the components, there are 11 major categories or patterns of cloud computing technology:

1. Storage-as-a-service
2. Database-as-a-service
3. Information-as-a-service
4. Process-as-a-service
5. Application-as-a-service
6. Platform-as-a-service
7. Integration-as-a-service

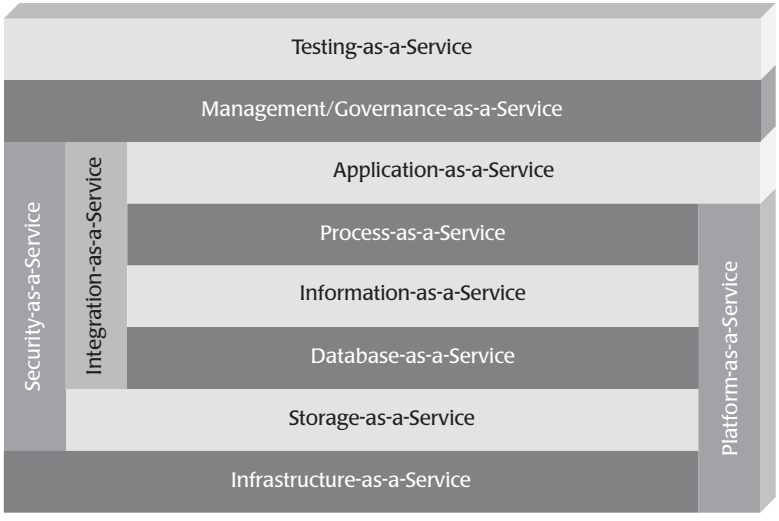

Figure 1.2 The components of cloud computing make up a wide range of services you can leverage over the Web through a subscription arrangement. Most services that you can leverage from a data center can now be leveraged from the cloud.

8. Security-as-a-service
9. Management/governance-as-a-service
10. Testing-as-a-service
11. Infrastructure-as-a-service

We go into more detail in Chapter 3, "Defining Clouds for the Enterprise," but it is useful to define them at a high level here.

Storage-as-a-service (also known as disk space on demand), as you may expect, is the ability to leverage storage that physically exists at a remote site but is logically a local storage resource to any application that requires storage. This is the most primitive component of cloud computing and is a component or pattern that is leveraged by most of the other cloud computing components.

Database-as-a-service (DaaS) provides the ability to leverage the services of a remotely hosted database, sharing it with other users and having it logically function as if the database were local. Different models are offered by different providers, but the power is to leverage database technology that would typically cost thousands of dollars in hardware and software licenses.

Information-as-a-service is the ability to consume any type of information, remotely hosted, through a well-defined interface such as an API. Examples include stock price information, address validation, and credit reporting.

Process-as-a-service is remote resource that can bind many resources together, such as services and data, either hosted within the same cloud computing resource or remotely, to create business processes. You can think of a business process as a meta-application that spans systems, leveraging key services and information that are combined into a sequence to form a process. These processes are typically easier to change than are applications and thus provide agility to those who leverage these process engines that are delivered on demand.

Application-as-a-service (AaaS), also known as software-as-a-service (SaaS), is any application that is delivered over the platform of the Web to an end user, typically leveraging the application through a browser. While many people associate application-as-a-service with enterprise applications such as Salesforce SFA, office automation applications are indeed applications-as-a-service as well, including Google Docs, Gmail, and Google Calendar.

Platform-as-a-service (PaaS) is a complete platform, including application development, interface development, database development, storage, testing, and so on, delivered through a remotely hosted platform to subscribers. Based on the traditional time-sharing model, modern platform-as-a-service providers provide the ability to create enterprise-class applications for use locally or on demand for a small subscription price or for free.

Integration-as-a-service is the ability to deliver a complete integration stack from the cloud, including interfacing with applications, semantic mediation, flow control, integration design, and so on. In essence, integration-as-a-service includes most of the features and functions found within traditional enterprise application integration (EAI) technology but delivered as a service.

Security-as-a-service, as you may have guessed, is the ability to deliver core security services remotely over the Internet. While the typical security services provided are rudimentary, more sophisticated services such as identity management are becoming available.

Management/governance-as-a-service (MaaS and GaaS) is any on-demand service that provides the ability to manage one or more cloud services. These are typically simple things such topology, resource utilization, virtualization, and uptime management. Governance systems are becoming available as well, offering, for instance, the ability to enforce defined policies on data and services.

Testing-as-a-service (TaaS) is the ability to test local or cloud-delivered systems using testing software and services that are remotely hosted. It should be noted that while a cloud service requires testing unto itself, testing-as-a-service systems have the ability to test other cloud applications, Web sites, and internal enterprise systems, and they do not require a hardware or software footprint within the enterprise.

Infrastructure-as-a-service (IaaS) is actually data center-as-a-service, or the ability to remotely access computing resources. In essence, you lease a physical server that is yours to do with as you will and, for all practical purposes, is your data center, or at least part of a data center. The difference with this approach versus more mainstream cloud computing is that instead of using an interface and a metered service, you have access to the entire machine and the software on that machine. In short, it is less packaged.

The Dream Team of Cloud Computing and SOA

While you can certainly leverage a cloud without practicing SOA, and you can leverage SOA without leveraging cloud computing, the real value of cloud computing is the ability to use services, data, and processes that can exist outside of the firewall in SEDC (somebody else's datacenter). Those who attempt to toss things to the clouds without some architectural forethought will find that cloud computing does not provide the value. Indeed, it could knock you back a few steps when considering the risks and cost of migration.

There will be some core patterns of success with cloud computing over the forthcoming years. Those who leverage cloud computing within the context of an architecture will succeed, while those who just toss things into the clouds as they think they need to will fail. Remember, SOA can provide a compelling business proposition when combined with cloud computing and an enterprise that needs this type of solution (see Figure 1.3).

Indeed, one can consider cloud computing the extension of SOA out to cloud-delivered resources, such as storage-as-a-service, data-as-a-service, platform-as-a-service—you get the idea (see Figure 1.4). The trick is to determine which services, information, and processes are good candidates to reside in the clouds as well as which cloud services should be abstracted within the existing or emerging SOA. We take you through that process in Chapters 4 through 11.

Simply put, you can think of clouds as additional places to run things. The advantage is that you do not have to drag yet another software-rich server into the data center along with the people required to maintain it.

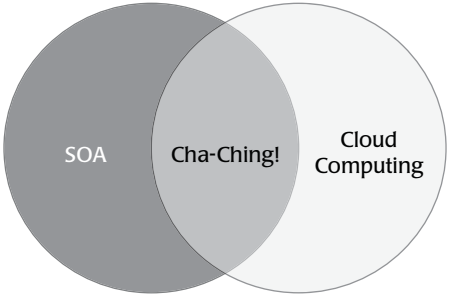

Figure 1.3 SOA and cloud computing provide a great deal of value when they work together.

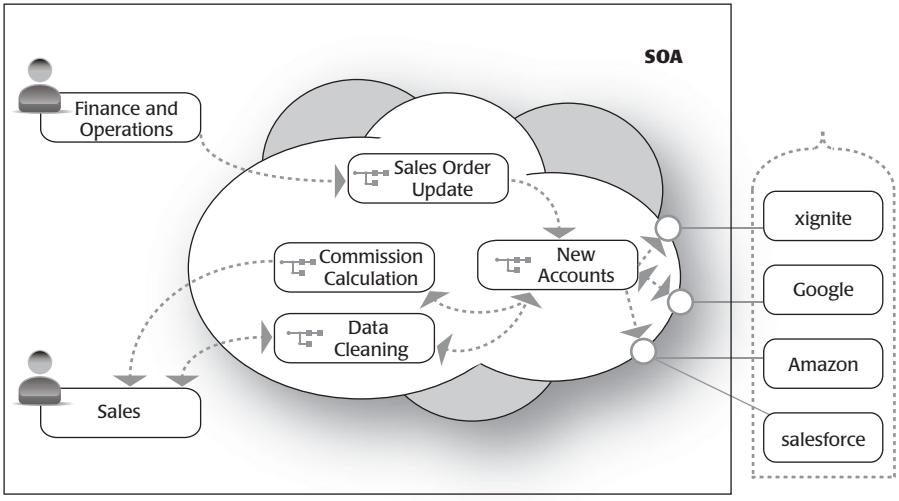

Figure 1.4 SOA can leverage cloud computing resources as services and use them as if they were contained within the SOA.

While enterprise IT is understandably skittish about cloud computing, many of the cloud computing resources out there will actually provide better service than on-premise utilities, once we allow cloud computing to settle in a bit more. Cloud computing benefits will continue to cushion the settling-in process, including cost savings, efficiencies, and access to thousands of dynamic Web-delivered resources.

Interest in cloud computing is also driving an interest in SOA. SOA not only is a mechanism to drive more reuse and agility but also offers the ability to figure out what should stay local and what should find a place in the clouds. Good SOA leads to a good cloud computing strategy, which leads to reduced costs, enhanced agility, and more excitement around enterprise computing than we have seen in awhile.

What SOA Can Learn from Cloud Computing

Service Design

Those who deploy services in the cloud, such as Amazon, Force.com, and others, have done a pretty good job with service design. You really *must* do a good job to rent the darn things out. Many SOA projects have a tendency to build

in services that are too course-grained, too fine-grained, or just not at all well designed. We discuss this issue in more detail later in the book when we talk about service design and modeling for our SOA using cloud computing.

In reality, unless services are not well defined and well designed, they will not sell well when delivered on demand. Those who provide services out of the cloud—which are most major cloud computing providers—therefore must spend a lot of time on the design of the services, including usability and durability. We urge those who build services within their SOA, no matter the enabling technology and standards involved, to look at the existing services available for rent as good examples of how services should be designed, developed, and deployed.

Service Expandability

Cloud computing services are designed to expand as needed, and those who leverage cloud services do so because they can get the services on demand, when they need them. The ability to expand services within an SOA is typically a painful and expensive process.

The fact is that services designed and developed within enterprises typically are not designed to scale. Indeed, the core issues with SOA revolve around the fact that many within IT do not focus on scaling until it is too late and too difficult to fix. Cloud computing providers had to figure out scaling rather quickly.

What Cloud Computing Can Learn from SOA

Service Governance

There is little notion of governance today within cloud computing, and thus there is little control and implementation of policies. Therefore, many enterprises are not diving right into cloud computing.

Governance, while not always well implemented, is a fundamental fact of life with SOA. The ability to set policies around services and to manage changes to those services is a critical success factor. As we weave cloud computing–delivered services into applications and within our SOA, we will find that many things break as the on-demand services change over time. Typically, SOA can manage the changes through SOA governance systems, but perhaps some of that governance should originate with the services that come out of the clouds.

Driving from the Architecture

Doing SOA properly means driving SOA from the architecture to the technology. Within the world of cloud computing, the resources on demand are the starting point. With cloud computing, the need for a well-thought-out architecture is just as important as for traditional systems, considering that you are extending the architecture out of the firewall.

Using cloud computing resources is about extending your architecture out of the enterprise to incorporate cloud resources, and thus it is important to remember that your architecture does not end at the firewall. Understanding both the resources that exist within the enterprise and the resources that are cloud-delivered is even more critical, as is the need to configure these resources correctly in the context of an architecture and to meet the needs of the business.

Clearly, SOA and cloud computing go hand in hand. Cloud computing is just the ability to leverage new platforms and resources that you do not happen to own. Nothing really changes outside of that, including the need to do SOA right. However, cloud computing is accelerating the adoption of SOA by providing aspects of SOA on demand. SOA can learn a lot from the clouds, and the clouds can learn a lot from SOA. This book brings the two together.

Making the Leap

If you purchased this book, clearly we do not have to sell you much on making the leap to the clouds. The trick is to make sure you leap in the right direction. There are a few things to remember before you embark on the journey of extending your enterprise architecture to the cloud and to keep in mind as you read this book.

First, the switch to cloud computing is not about a quick fix; it is about moving your IT architecture incrementally forward, leveraging approaches around SOA and cloud computing resources when they make sense. Those who want to drive to changes in IT quickly and tactically will find useful information here, but a bit of architectural planning is strongly recommended. So, get excited, but stay on topic.

Second, look at the people and process issues within your enterprise at the same time you look at the technology. Many technologists miss this part and end up doing a very good job of driving toward a new architecture, but if

nobody accepts it or pays for it, it is all for naught. Those who are successful with any systemic architecture change take into consideration the people and cultural issues.

Third, make sure to define the business case. We believe so strongly in defining the business case that we devote Chapter 4 to this topic. IT professionals need to get in the habit of working from the business to the architecture and then to the technology. All changes to the existing information systems should be justified as a clear business case, and the IT team should have to sell this change to the stakeholders and sponsors within the organization. If the change does not ultimately bring value to the bottom line, it should not be done.

Finally, do not get caught up in the hype—at least not too caught up. It is not productive when conversations around SOA and cloud computing degrade into debates around technology or standards before the problem is even clearly understood. We love to do that because we love technology.

Being Positively Disruptive

The idea is to drive change for the better. Many enterprises are in such bad shape that it makes sense to leverage disruptive technology and approaches to drive that change. This is really about rethinking, redefining, and shaking things up. The use of SOA using cloud computing is analogous to the first movement to the Web years ago. The use of the Web revolutionized the way we access and view information, and cloud computing will revolutionize the way we look at IT resources. It is about making major disruptive changes to very poorly planned IT infrastructures that drive changes for the good.

You must keep in mind as you read this book that the shift to cloud computing is all about change for the better using intense approaches and technologies that make sense. It is also about changing hearts and minds about adapting these technologies and approaches, and in essence, that is the most difficult part of the journey.

Many of you will face resistance from people in your organization who are reluctant to support the change. While many who drive disruptive change tend to view their reluctance as a hindrance, it is really an opportunity for you to test your ideas and learn how to explain them. Testing your ideas means listening to the points made by those resisting the change and using those points to see if you have missed anything in your assessment of what

must be changed and why. While the naysayers may be overly negative at times, you need to welcome the opportunity to review your approaches and perhaps make revisions based on their feedback. You will have a firmer belief that what you are doing is right and perhaps will gain new knowledge and insights in the process.

Your role as teacher gives you the opportunity to learn how to educate people about what you are doing, and why. Those who are most successful in driving positive disruptive change are those who can thoroughly and with conviction explain the value of the new approach and new technologies.

The benefit for you is that if you successfully shift your company to a cloud computing and SOA architecture, your enterprises will be much more effective and efficient, able to meet most, if not all, of the needs of the business. You will have a key competitive advantage that allows you to increase your market share, build better products, and live up to the mission of the organization. You will have a healthy IT infrastructure that enables you to do more with less.

This page intentionally left blank