# Chapter 7
# Deployment Architecture

Enterprise-class software systems have seen many phases of evolution. Centralized mainframe systems evolved into client/server systems, client/server systems evolved into distributed systems, and distributed systems, still in their infancy, are now being recast as reconfigurable Web services. Each of these deployment architectures spawned many variants. Emerging Web architectures extend these ideas and add new ones, bringing alleged new benefits for users and keeping both marketects and tarchitects busy. Not surprisingly, we are also carrying each of these major architectural styles into the future, as much for the unique advantages they offer as for the legacy systems they've left behind.

The wide variety of deployment architectures for enterprise-class software is starting to more strongly influence deployment architectures for common packaged software applications ("shrink wrapped" applications that usually cost less than $500). We're now seeing traditional applications offered as services. As bandwidth continues to increase and hardware devices become more sophisticated, the number of deployment choices increases.

I use the term *deployment architecture* to describe the manner in which a customer deploys a system. This is related to the UML definition of deployment architecture but my focus is more on the strategic implications of a deployment choice and less on the lower-level decisions such as how to allocate work in a multiprocessor computer system. Emerging Web technologies and business models present tarchitects and marketects with considerable creative flexibility. This chapter will help you sort through some of the business and technical issues associated with choosing a deployment architecture so that your customer will see your choice as a winning solution.

## Deployment Choices

Common choices for software deployment architectures are discussed in the following sections.

### Customer Site

This type of deployment is the most traditional and the most common. The software is installed at the customer's site and is configured, operated, and maintained by the customer. For common packaged software for the consumer market, such as I'm using to write this book, it means I've purchased, installed, and configured it on my laptop. Enterprise-class software, such as corporate finance system, warehouse management, and customer relationship management (CRM), is almost always controlled by the corporate IT department.

Enterprise-class software is usually installed and configured through a consulting assignment. The system vendor may provide professional services to the customer or subcontract them to a major consulting firm (such as EDS, Bearing Point, or Accenture). The duration of the assignment is often a function of the system's effect on existing corporate processes, any or all of which may have to be substantially modified before, during, and often after the installation.

### Application Service Provider

An application service provider (ASP) operates an application for the customer, offering a limited set of services, and rarely a complete solution. For example, the ASP may provide "24/7" system monitoring, routine backups, and automatic access to additional internet bandwidth, but not application technical support. The specific services provided by the ASP must be negotiated.

ASPs are slightly more common for business applications, although that is changing. For example, many early ASPs offered large enterprise applications to smaller customers. Newer ASPs offer consumer applications, such as estate or tax planning software, as a service.

I'm not making a distinction between a software publisher that offers its application as an ASP and a software publisher that licenses its application to a third party that then offers it as an ASP. These distinctions, while important for the publisher and the third-party provider, are largely irrelevant for the purposes of this chapter.

### Managed Service Provider

A managed service provider (MSP) extends the concept of an ASP by offering an array of services in addition to the operation of the application. In marketing terms, an

ASP competes at the level of the generic/expected product while an MSP competes at the level of the expected/augmented product. The exact services offered vary, but usually they include extensive monitoring and backup, and possibly call center support, commerce operations/management, and security/firewall management. Early MSPs targeted the business market. Some recent ones target very specific consumer markets, such as corporate e-mail hosting or digital photo appliances.

MSPs focused on very well-defined specialized hardware market niches will continue to emerge. For example, in the financial services industry some very large MSPs offer online banking services to millions of consumers. Of course, you don't know this because the MSP has allowed the bank offering these services full branding control.

In both ASP and MSP, relationships it is common for service level agreements (SLAs) to precisely define acceptable performance parameters. Also, when the application is a traditional software system, at least some customer data is stored at the service provider. This has important tactical and strategic implications for security and operations that extend far beyond the service actually being offered.

### Transactional (Web Service)

A transaction deployment is one that computes an answer in a single, whole transaction, often through Web service protocols. It commonly provides services to individual users, such as when you ask for a map on the Internet. In certain cases end user data may be stored at the service provider, but this is rarely corporate data. This style of system is not yet common in enterprise software, but recent efforts to build complex systems around collections of transactional Web services may dramatically increase its use. Web-service based application architectures will eventually become common for every type of application. This does not mean that they will "win," because they are not appropriate to every market segment. It does mean that we are going to be faced with increasingly sophisticated choices.

The four broad categories just outlined capture the basic deployment options. Savvy marketects have created subtle variants to gain a competitive or positioning edge. For example, some service providers classify themselves as "Internet business service provider" (IBSPs), focusing on a single element of a complex solution (e.g., loan portfolio management for banks). Others define themselves as enterprise application providers (EAPs) and focus on a single kind of enterprise-class system, in the hope that they can leverage that experience across their entire customer base. In the framework presented above, IBSPs and EAPs would be classified as either managed service providers or application service providers, depending on the specific services they offer.

### The Hybrid Deployment Architecture

I've been presenting deployment architectures as relatively exclusive choices: either ASPs or MSPs. In fact, there is no technical requirement that the deployment architecture be all one way or another. As is true with other aspects of tarchitecture, it is often best to let your customer and the problem guide you in your choice.

One of the more successful tarchitectures I helped create was a true hybrid. In this case, customers needed realtime access to more than five terabytes of data and required certain kinds of private transactional data. The solution was straightforward: Put some parts of the design, such as the transactional data, at the customer's site; put other parts, such as the five terabytes of data, at a service provider. Making this hybrid architecture work was a bit of a challenge, but the benefits to our customers were worth our efforts.

## Customer Influences on Deployment Architectures

Choosing a deployment architecture that meets your customer's expectations is an important part of a winning solution. Consider the following customer concerns when making this choice.

### Control and Integration

Customers vary in their perceived need to *control* the software system. As a reasonably sophisticated end user of software, I want to control what extensions or changes I make to my system. Thus, I don't want a managed service for my laptop because I want to be able to add or remove software as I see fit. But I don't want to control everything! I just want my high speed internet connection firewall to "work" without spending a lot of time configuring or adjusting it.

Customers of enterprise-class software have similar demands for control, expressed in a number of ways. They may want to control the system's operational infrastructure. Specifically, they may believe that the control afforded by a system operating onsite is absolutely crucial for mission-critical applications ("I know *exactly* who is going to answer the pager if the system breaks and how long it will take them to fix it"). Conversely, they may be happy to give control to a service provider based on their perception that the service provider can provide high-quality, uninterrupted service.

Another issue associated with control is the long-term retention and management of data. Service providers who approach this topic carefully may be able to argue that they can do a better job than the customer. Alternatively, the customer may already have sophisticated policies, and may simply feel more confident in their own ability to manage data.

It is imperative that the marketect understand the control issues that are most important to the target ASP or MSP customer. Failing to do so will prevent you from creating a winning solution. Conversely, understanding the deeper concerns that your target customer has regarding deployment will enable you to handle them with skill.

As *integration* needs increase so does the likelihood that the system will be deployed at the customer site. This can range for integration between common desktop applications to linking your CRM system with your inventory management and fulfillment system.

## Data Security/Privacy and Peak Loads

The nature of the data manipulated by the system influences customer perceptions of an appropriate deployment architecture. Hospital payroll data, for example, may not be viewed as sensitively as patient records. As a result, the hospital may opt for a managed service solution for payroll processing but license patient record management from a reputable vendor and run it internally. A professional services firm, on the other hand, may have a strong need for privacy with respect to payroll data and require that it be managed inhouse. In the case of my home firewall, there is no data to manage, so there is no need for data security. Nonetheless, I purchased a hardware-based firewall because I didn't want to pay a monthly service fee to my ISP. You need to understand your customer's relationship to the data managed by your application, as developers can easily make poor choices based on faulty assumptions. I will talk more about data later in this chapter.

Depending on the application, it can be more cost-effective to place the application at an ASP/MSP whose equipment and/or communications bandwidth can handle peak loads that are not cost-effective to handle with your own dedicated equipment.

## Costs and Vendor Confidence

There are times when an xSP (meaning either an ASP or an MSP) can offer a sophisticated, expensive application at a fraction of what it would cost the customer to license and deploy it onsite. Sometimes this is because the xSP can sublicense an application in way that provides access to smaller businesses. Other times it's because the customer doesn't have to invest in the total infrastructure needed to make the solution work.

How does your customer perceive you? Are you a stereotypical Internet startup with programmers sleeping on cots, or are you an EDS- or IBM-style systems integration firm where formal dress is still common? Put another way, would you trust your company's most sensitive data to an Internet startup that can't properly manage its internal servers' passwords? Answering these questions will help you understand why a customer may be somewhat reluctant to entrust the operation of its mission-critical system to your company's engineers.

Earlier I mentioned that failed ASPs have made customers wary about trusting their data and operations to outside parties. Be aware that your promotional materials

may have to move beyond simply touting the benefits of your solution to providing access to detailed financial statements in an effort to demonstrate that you have long-term viability in the marketplace. This is often referred to as a *viability test,* and unless you can demonstrate that your company will be viable for the next few years, you may not win the deal.

As stated earlier, another aspect of confidence concerns the manner in which you archive and retain your customer's data. As your total solution evolves, you will inevitably change the kind of data you're storing. Reassuring customers that they can always get at old data is important to your complete solution design.

## Customer Skills and Experiences and Geographic Distribution

Your application and its deployment architecture will dictate the skills and experience required by your customer's staff. If the application has relatively simple operational requirements, this isn't really a factor. It is when the application exhibits complex operational requirements that the choices become interesting.

The easiest deployment choice, from the perspective of the vendor, is to make your customer responsible for all system operation. They have to care for and feed the application—monitoring it, backing it up, administering it, and so forth. All of these are skills that must be learned.

If you elect to offer the application as an xSP or license an xSP to offer the application on your behalf, you or your xSP partner will have to assume the responsibility to meet the needs of your customer. Be careful, as customers often place greater demands on xSPs than on their own MIS staff. One or the other of you will have to hire a staff with enough experience and skills to create the necessary data center. You can subcontract this, and many companies do, but you still need someone with the necessary experience to manage the subcontractor.

It is important to consider the operational culture associated with the deployment architecture. If your company started as a highly dynamic, "let's just get it done" entrepreneurial venture, you may find it hard to switch to the more formal demands of data center operation. In an entrepreneurial company, for example, data center security is often lax; in an xSP, data centers require tight operational control.

It can often be easier for a customer to provide an application to geographically dispersed employees or workgroups when an xSP manages it.

As you consider how customers influence your choices, keep in mind that your target customer will provide you with the strongest requirements for a deployment architecture. Earlier in the book I talked about the dangers of resumé-driven design, in which designers make technical choices in order to pad their resumés. I've witnessed the same phenomenon in choosing a deployment architecture. Investor-driven design swept through Silicon Valley in the late 1990s as many startups were capitalizing on the growing popularity of xSPs. Many of these xSPs subsequently failed, for reasons that included an inability to understand their customer's true motivations for a deployment architecture. Unfortunately, some failures were truly catastrophic. Many customers

### Please Disregard Those Credit Cards

One of my clients running a managed service had a rather embarrassing episode when the director of managed services operations distributed a spreadsheet that detailed the growth in customer transaction revenues. Unfortunately, he forgot to delete the worksheet that contained the complete customer contact information, including credit card numbers! As you can guess, much more appropriate operational controls were put in place after this incident. As the vendor, you should make certain this kind of mistake never happens. As a potential customer of an ASP or MSP, you have the right to demand a careful review and audit of all operational procedures.

irretrievably lost crucial corporate data when these companies went under. Such losses have made corporations justifiably wary about storing key data at a service provider.

For this reason, and others, it is essential that the marketect choosing a deployment architecture work to understand not just customer requirements but corporate objectives as well.

### When the Risk Really Is Too Great

One client of mine had created a traditional, enterprise-class system for manipulating extremely sensitive data that represented hundreds of millions of dollars worth of intellectual property. The initial customer-site deployment model was a success because customers were allowed complete control over its every aspect. Unfortunately, it came with a rather high price tag, and my client felt that they had to offer an ASP model to broaden their market. Unfortunately, they failed to understand that the factors described above are not isolated from each other. Instead, each is considered and weighed against another in the context of a winning solution.

In this specific example, the extreme sensitivity of the data meant that my client would have to build a substantial infrastructure and ensure that the data center was operated under exceptionally stringent conditions. However, they had neither the organizational nor operational maturity to undertake this task. In addition, many of the target customers were global companies who required 24/7 access and support. Preliminary research also indicated that customers didn't want an ASP—they wanted an MSP. Sadly, my client chose not to invest in the additional infrastructure to create an MSP until the new model had proven successful.

I urged my client to avoid offering the ASP and to create a truly winning solution. To my dismay, they went ahead with their plans and introduced the new model. It failed, and the company eventually went bankrupt.

## Corporate Influences on Deployment Architecture

The marketect must consider more than customer requirements when choosing a deployment architecture. Sustainable winning solutions require the marketect to understand the capabilities, desires, needs, and short- and long-term strategies of the corporation offering the solution. Here are some of the strongest corporate influences on deployment architectures.

### Sales Cycle

The sales cycle refers to the length of time and number of steps it takes a corporation to make a sale. In general, it is correlated to the price and complexity of the software. Consumer software, with low price points, simpler implementation and integration requirements, and very few decision makers (usually one or two) has a relatively short sales cycle. Enterprise-class software, with substantially higher price points, complex implementation and integration requirements, and many decision makers (usually a committee with or without approval authority) usually has a longer one. When a corporation is considering a multimillion dollar purchase, they're going to think about it carefully. I've been involved with sales that took two years or more.

In general, most consumer software is still deployed on a PC (a customer site deployment), so I won't discuss its relationship to the sales cycle further. Business software is a different matter. If you seek a shorter sales cycle, consider deploying your software as an xSP or licensing it to one that is properly qualified. The sales cycle is usually shorter and the implementation cycle should be. This is important, as once customers have made the commitment to your product they're going to want it up and running as quickly as possible. Watch out, though. Choosing this option without understanding customer influences is not going to create a winning solution.

Experience with xSPs indicates an even more complex situation than I've just described. There are times when a customer wants to use an xSP as a quick starter solution and then migrate it to their site (for an on-site deployment). I've also had to manage reverse migrations, in which customer-site deployments were migrated to a service provider primarily because of the service provider's superior infrastructure. While this kind of deployment migration is extremely rare at the moment, I expect that it will become more common in the future.

### Infrastructure Investment

When an application provider considers offering their solution as an xSP or Web service, they must carefully consider the investment needed to create a long-term, viable offering. Companies with a service-based applications model routinely underestimate the often significant investment that is required to create a reliable infrastructure capable of handling projected demands.

You don't have to create a solid infrastructure, but then you risk poor customer service. Investment calculations must include technical resources (such as hardware and infrastructure) and nontechnical resources (experienced data center staff, support staff, and so forth). Unless your corporation has the necessary capital and willpower to invest in a solid infrastructure, I recommend against an xSP or Web service.

### Cash Flow

Like sales cycle and infrastructure investment, cash flow must be carefully modeled. Suppose, for example, that an MSP offers a complex enterprise application to customers on a rental basis, with no minimum contract. If they are paying an annual license (common), they have a single payment due at the beginning of their year of service. If their customers are paying a rental, it may not be enough for the MSP to pay for the next annual license. Ultimately, xSPs must take a careful, sober approach to managing the cash reserves needed for successful long-term operations.

### Flexibility

An installed base of customers who use your software on their premises can limit your ability to rapidly innovate and improve it. Customers who have invested time and money in a given release are usually reluctant to modify it. As a result, chances are good that if you go with customer-site deployment you will be supporting several releases in the field simultaneously.

This is one of the most appealing aspects of offering your solution as an xSP or Web service. By maintaining complete control, you gain considerable flexibility in such things as upgrade schedules. In an emerging market, where rapid release cycles are often required for growth, you have the luxury of upgrading as often as necessary. Patches can be quickly obtained and installed by development. Of course, appropriate care must be taken whenever you modify your own operational environment, but installing a new release in an operational environment that you control is usually much easier that coordinating upgrades across dozens, hundreds, or thousands of customers.

### Geographic Distribution

If you're trying to sell an application to a global company, they have the right to expect that the deployment will provide them with the necessary support and service. When they choose a customer-site deployment, they can control the level of service provided by their inhouse MIS staff, including such things as the language used to answer the phone. When considering an xSP, they may require that it provide local customer support throughout the world.

A global company may also make unforeseen technical demands on the total solution. If an application is deployed inhouse, the customer is responsible for installing

and maintaining it as appropriate, subject to the license terms. If, on the other hand, you're providing the application as an xSP, you may have to install it in multiple locations around the world to guarantee such things as performance and availability. Communications networks are fast and increasing in speed every day, but for the most part locally maintained applications and data are faster.

### Service, Not Price

Early xSPs competed on price. Most of them have not survived. Second- and third-generation xSPs have begun to compete on service—convenience, reliability, support and so forth. They have a chance to make it in the long run, provided they maintain their service focus.

## Choosing a Software Deployment Architecture

Figure 7-1 illustrates the rough relationships that exist between customers, corporate influences, and deployment architectures. It also captures the effects that one dimension often has on the others. For example, if your enterprise-class customer has a high need for integration with existing or legacy systems, wants more control over system operations, and is dealing with highly sensitive data, chances are good that they will favor a system that can be deployed at their site under the control of their MIS staff. Relaxing these constraints makes other deployment choices viable and in many cases even preferable.

I've intentionally simplified many of the variables that are associated with deployment architecture in the figure. For example, even if the customer has a high preference for deploying the system on site, they may still choose an ASP/MSP if the skills of their operational staff are insufficient or if the application's peak demands will exceed inhouse capabilities. The variables in black—control, integration, and data—all share the same effect on the choice of deployment architecture (when high, deploy at the customer site). The variables in gray are on an opposite scale—when high, deploy as an ASP, MSP, or Web service.

The upper right corner of Figure 7-1 captures some of the corporate influences previously described. The figure also shows the migrations between various deployment options as light gray lines.

## Deployment Architectures and the Distribution of Work

No matter what deployment architecture you choose, someone is going to have to install, maintain, administer, and support it. These basic service and support functions can't be ignored. In a customer-site deployment, or when integrating with a Web service,

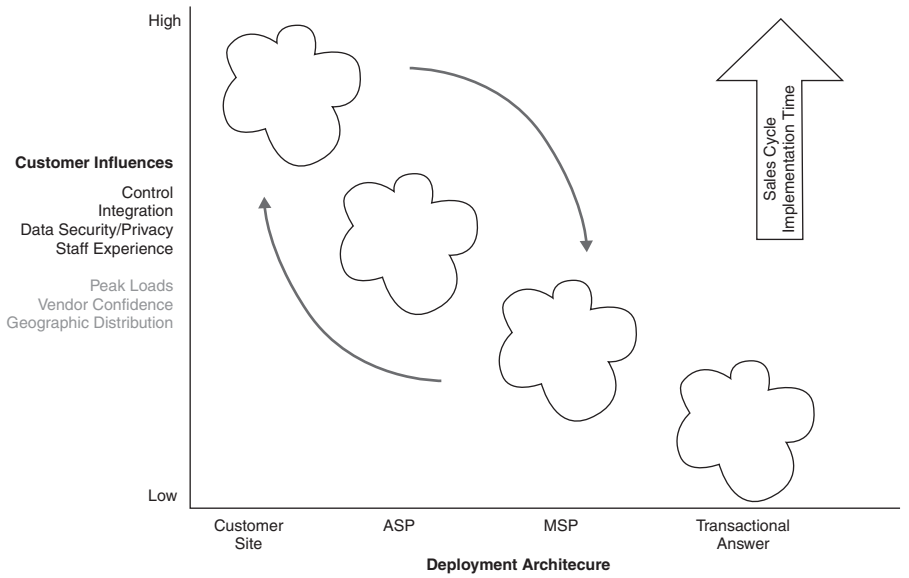


**FIGURE 7-1** Choosing a deployment architecture

the customer is responsible for the bulk of the workload. In an ASP, these responsibilities are shared. In an MSP and transactional service, they are the service providers. The various deployment choices shift the locus of control among the entities, as shown in Figure 7-2.

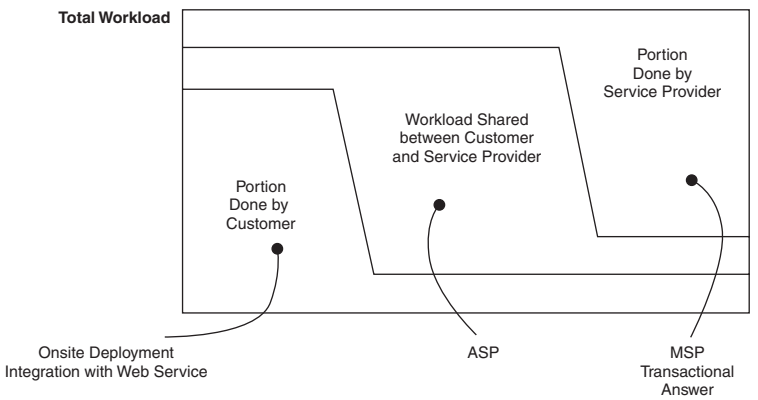


**FIGURE 7-2** Distribution of work in deployment architectures

## The Information Appliance

A completely different kind of deployment architecture, and one that can be operated almost equally well at a customer site, as an ASP, or as an MSP, is the information appliance—a specialized device that provides one or more specific functions designed to create a better total solution for a well-defined target market. Examples of information appliances include

- Linksys EtherFast DSL Firewall/4-port hub. I have installed this in my home to provide me with a simple firewall and 4-port LAN.
- Aladdin Knowledge Systems, Inc., eSafe appliance. This provides anti-virus and malicious mobile code protection for enterprises.
- TiVo and Replay digital video recorders. These provide a single device for managing television viewing.

Several trends motivate the growth of information appliances. One of the biggest is the continued adoption of the Linux operating system, which is reliable, free from expensive licensing models, and easily customized and embedded. While there are other excellent choices for appliance operating systems, most of them come with a license fee, which directly increases total cost but provides little additional value to users.

One point in favor of proprietary operating systems is any tools they have created to support a specific market niche, making them better than similar tools available on Linux. In other words, there is no simple way to make the choice; you will have to compare a full set of issues, among them license fees, development tools and related infrastructure, and development team experience. What is undeniable is that Linux is the platform of choice for a growing number of information appliance vendors.

Another important trend is the need to simplify complex solutions. Appliance vendors usually place a premium on simplicity. Just install the appliance and set a few simple parameters. Many times you don't have or need things like keyboards, monitors, or even expansion slots. The absence of such items simultaneously drives down costs and simplifies use.

Information appliances are not appropriate for every kind of software, particularly any system that creates or manages data or that requires substantial customization or programmatic integration, or that just runs better on existing hardware. That said, the move toward Linux and other open source software will continue, as will our desire to reduce complexity for our customers. Thus, we will see the continued growth of single and multi-function information appliances.

## Deployment Choice Influences on Software Architecture

A given deployment choice may exhibit any of the following influences on your software architecture.

### Flexible, Parameterized, or No Integration Options

A system deployed at a customer's site has the greatest demand for flexible integration options. A system deployed as an ASP or MSP may require flexible integration, but the xSP vendor is often not motivated to provide it because of its extremely high costs. In fact, the more standard the xSP can make the offering, the better. Standardized offerings are simpler to create, simpler to manage, and more profitable to operate. Any deployment choice can be offered with no integration options, which is surprisingly appropriate when you're creating a system with very well defined functions and relatively simple boundaries.

### Upgrade Policies

Different deployment architectures make different upgrade demands. For systems deployed at a customer site upgrades must be carefully planned to be minimally disruptive. This can be especially challenging if the customer has crucially important data or has extensively integrated the production system with a large amount of their own programming. For this reason, enterprise-class systems deployed onsite are rarely upgraded more than once every nine months. In contrast, I know of one world-class MSP that safely modifies their production system approximately every 10 to 12 weeks, rapidly introducing new features to their customers in an emerging market. They are able to do this because early in the development of the system they made substantial changes to make upgrading easier. Upgrades are discussed more thoroughly in Chapter 12.

### Data Protection and Access

Application data maintenance must be appropriate based on the application, the users, and the data's sensitivity/importance. When the system is deployed at a customer site all responsibility for handling these issues, especially as they relate to corporate data, is the customer's. As mentioned earlier, the converse is also true: Storing customer data at, or as, an xSP requires that the xSP or your engineering, product development and operations staffs follow strictly defined guidelines that deal with proper data handling. Would you let your company manage your confidential information?

### Migration Options

I expect an increase in the number of solutions that can be deployed either as ASPs/MSPs or on a customer-site. Furthermore, I suspect that these solutions will ultimately need to support migrations as previously described in this chapter. The possible effects of migration should be considered in the overall design of your architecture.

## The Future of Consumer Software

Web services enthusiasts paint a picture of the future in which savvy, Web-connected users won't license software for use on their home computer, but instead, will connect to a Web service via a persistent, reliable, high-speed Internet connection and license the software on a rental or subscription basis. Fortunately, the deployment architectures presented here, which are currently most applicable to enterprise-class software, will guide you in navigating this brave new Web services world.

Like enterprises, users will make their deployment choices based on the quality of the *solution*, not on the technology powering it. For people like me, who rely on their laptop during international travel, the thought of using software only via an Internet connection seems a bit crazy—perhaps in the future, but certainly not now. However, for many, accessing software via a trusted vendor through the Internet is very appealing. I'd like to see certain data, such as financial records, properly maintained for the rest of my life. Other data, such as my digital photos, I *want* to be shared, making a Web services model a natural fit. Because of these forces, and others that will emerge, I envision a complex environment for consumer software like that for enterprise-class software, creating very interesting choices and tradeoffs for marketects, tarchitects, and their customers.

❑ ❑ ❑ ❑ ❑ ❑ ❑ ❑ ❑ ❑

## Chapter Summary

- Your deployment architecture is the manner in which the system is deployed for use by a customer. Common choices include
  - Customer site
  - Application service provider (ASP)
  - Managed services provider (MSP)
  - Variant of a service provider (xSP)
  - Web services

  Hybrid models, in which part of the system is deployed at a customer site and part at a service provider will become increasingly common.
- Customer influences that motivate the selection of a deployment architecture include
  - Control desired
  - Integration with other systems
  - Data security/privacy
  - The ability to handle peak loads

  - Initial and ongoing costs
  - Customer confidence in you
  - Skills and experience of the system's operational staff
- Corporate influences on the selection of a deployment architecture include
  - Desired and actual sales cycle
  - Infrastructure investment
  - Financial model, most notably cash flow
  - Desire to move quickly and efficiently in managing your customer base
  - Geographic distribution of your company relative to your customers
- The choice of a deployment architecture does not change the total of work associated with a successfully managed system. It may change the distribution of this work.
- Information appliances are a growing category for deployment architectures in a wide variety of environments. Open-source licensing models, which can lower total costs of ownership, are in part fueling this growth.

### *Check This*

- ❑ Our deployment architecture matches our target market's need for
  - Control
  - Integration
  - Data security/privacy
- ❑ We have sufficient performance models and are sure that our deployment architecture can handle all anticipated workloads (see Chapter 10).
- ❑ We have instituted appropriate operational policies.
- ❑ We have accounted for the following in our choice of deployment architecture:
  - Sales model and sales cycle
  - Required infrastructure investment
- ❑ We have defined the amount of work we expect the customer to perform.

### *Try This*

1. Using Figure 7-3, identify how your software is deployed and the key forces behind this choice.
2. What would you have to do to change your solution from its current deployment to a new one? Would doing so enable you to expand your current market or allow you to reach a new one?

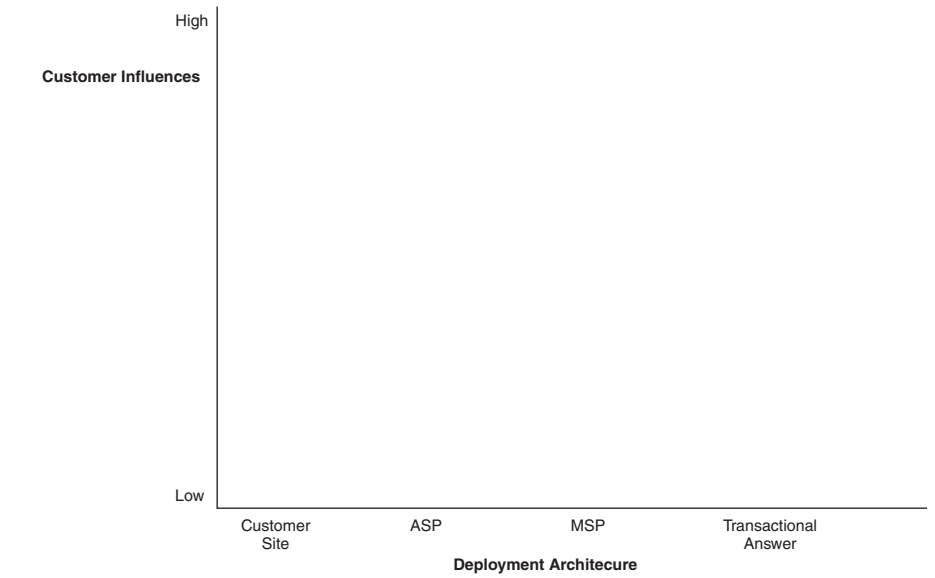


**FIGURE 7-3** Identifying your software deployment