SOFTWARE + SERVICES IN THE MICROSOFT WORLD

A TECHNOLOGY OVERVIEW FOR IT DECISION MAKERS

DAVID CHAPPELL

NOVEMBER 2007

SPONSORED BY MICROSOFT CORPORATION

CONTENTS

INTRODUCING SOFTWARE + SERVICES

In not much more than a decade, the Internet has remade large parts of the technology world. Despite these big changes, there are still many more to come. One of the most important is the shift within enterprises from relying solely on local software to a world of software plus services (S+S).

This overview describes S+S, focusing on what Microsoft is doing in this area. Yet don't be confused—while not everyone uses the same terms, the entire industry has embraced the idea of S+S under various headings. And while the move to S+S certainly affects consumers, this description focuses on S+S for business users.

S+S isn't a futuristic idea; it's a reality today. Yet the full impact of this transition is still years away. Understanding what that impact will look like requires thinking about the effect on applications—packages as well as custom software—and on the platforms those applications depend on. This overview describes both.

DEFINING TERMS

What does the term "S+S" really mean? The first "S" stands for software, of course, but the second "S" also really stands for software. All that changes is where that software runs and who runs it. Figure 1 illustrates this idea.

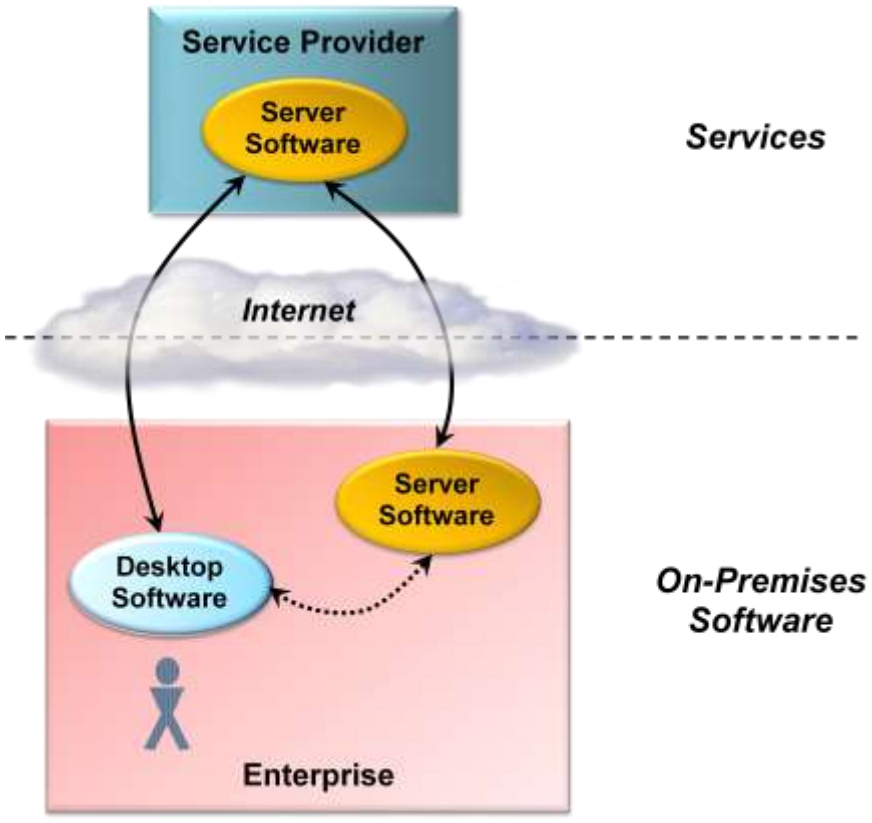

Figure 1: S+S means using both on-premises software and services provided by Internet-accessible software

As the figure shows, the desktop software that an enterprise user relies on might run solely on her desktop system. Alternatively, it might communicate with server software on another machine. If this desktop and server software is running within the user's enterprise, it's known as *on-premises* software in the S+S world. Don't be confused by this term—the software might actually be running in some other part of the organization rather than in the same physical location. What "on-premises" really means is that the software is running on an organization's own computers and in its own data centers. That organization is fully responsible for licensing, installing, operating, and maintaining this software.

Both desktop software and on-premises server software can also access functionality implemented by server software run at some service provider. This functionality is what's meant by the second "S" in S+S: software functions provided as *services*. These services might be provided directly to the user via a Web browser, to either desktop or server software through Web services, or perhaps in some other way. Whatever mechanism is used, because these services are accessed on demand via the Internet, they're sometimes referred to as being in the *cloud*. Yet in the end, services are just server software running on remote computers.

How services are implemented can vary. A service might be just an ordinary packaged application, licensed as usual from a software vendor, that's run by a service provider. Plenty of software is used this way today, including applications such as SAP or Oracle and infrastructure software such as Microsoft Exchange, Windows SharePoint Services, or Lotus Notes. A newer approach to the idea of providing software-based services relies on creating an application that's specifically designed to be deployed and used as a service. Especially with this option, exemplified by firms such as Salesforce.com, the customer typically pays based on its usage of the application—pricing is subscription-based.

Both of these approaches fit under the broad heading of *Software as a Service (SaaS)*. In fact, calling something a service really just means running it in the cloud, i.e., at some service provider. Given this, the real meaning of S+S becomes clear: The first "S" means on-premises software—software that's entirely operated by the organization that uses it—while the second "S" means SaaS.

Since both incorporate the notion of services, it's also useful to think about the relationship between S+S and service-oriented architecture (SOA). How these two ideas fit together depends on exactly what you think SOA is. One option is to view SOA as an architectural approach to building and connecting on-premises software, i.e., software running inside a particular enterprise. From this perspective, SOA is purely about exposing and using services within that enterprise—it doesn't include SaaS. A slightly wider view of SOA includes both services provided by an enterprise's on-premises software and services provided by the software of business partners, such as suppliers and customers. These partners aren't typically service providers, and so there's still a distinction between SOA and the idea of SaaS.

In the broadest view of SOA, however, services can be provided by an enterprise's on-premises software, by the software of business partners, and by software owned by service providers. Taking this more expansive perspective includes both SaaS and service-oriented interactions between on-premises software under the heading of SOA. While technically this view is hard to argue with—services are services, and the technology issues are much the same—defining SOA so broadly can hide important distinctions. Service providers have a business model that's typically quite different from what's applied in intra-enterprise services, for example. Still, especially in large organizations with cross-division chargeback mechanisms, providing internal services can look much like SaaS. From this perspective, S+S is the natural evolution of SOA, another step toward an increasingly service-oriented world.

Neither on-premises software nor SaaS are brand-new ideas. Given this, what's different about S+S? The answer is that looking at the world through an S+S lens is fundamentally different than taking either a pure software view or a pure services view. Going forward, both are likely to be important parts of an enterprise's software strategy.

S+S TODAY: SOME EXAMPLES

Some software, such as Internet search, should naturally be provided as a service. Yet much, maybe even most, software can potentially run either as on-premises software or as a service. And in some cases, the best results come from combining the two, with on-premises software augmented by services.

Email provides one clear example of the value that both software and services can provide. With Microsoft Exchange, for example, organizations today can choose various options, as Figure 2 shows.

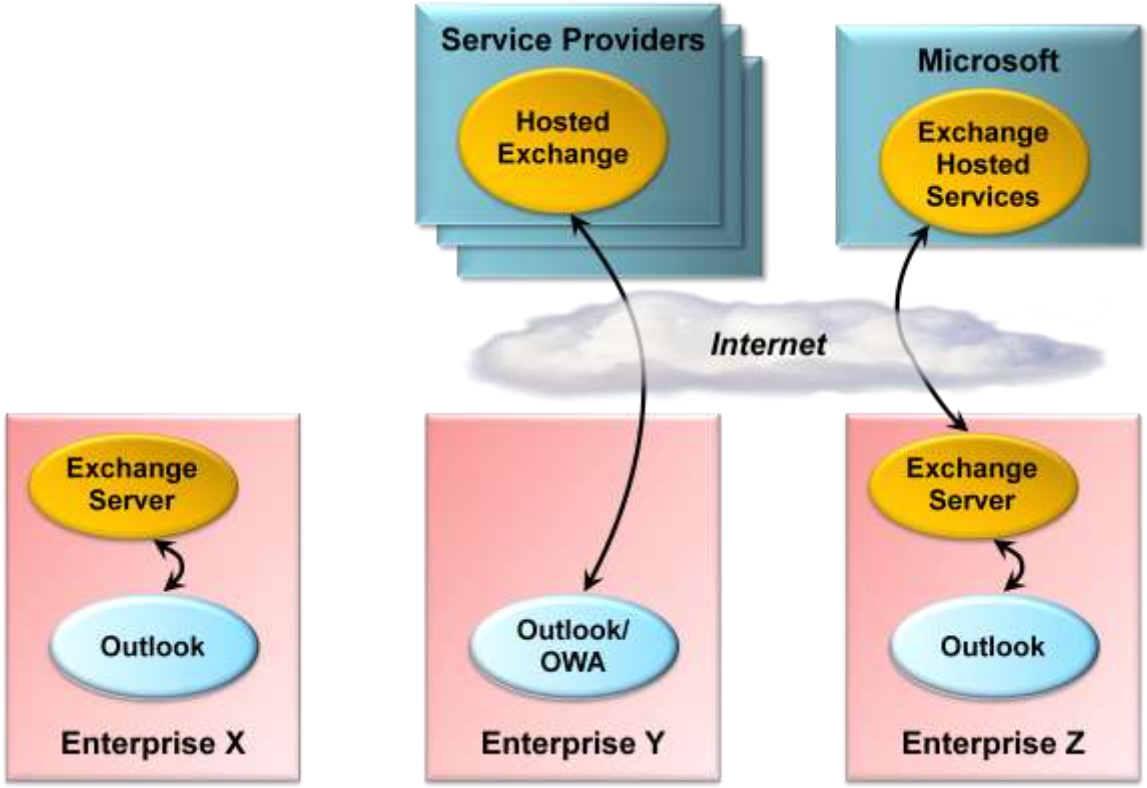

Figure 2: Exchange Server can be used as on-premises software, as a service, or as a combination of the two

In this example, Enterprise X has chosen to deploy Exchange Server as on-premises software. Enterprise Y instead uses a hosted Exchange service offered by Microsoft or by one of the many service providers in this market. This service might be accessed directly from Outlook clients, through a browser using Outlook Web Access (OWA), or in some other way. Enterprise Z combines both on-premises software and a service: It deploys Exchange Server itself, but also subscribes to Microsoft's Exchange Hosted Services. These services provide malware protection, email archiving, and other add-on services to this organization's on-premises Exchange Server. To distinguish between these two kinds of services, Microsoft sometimes describes a service such as hosted Exchange as *finished*, since people can use it

directly. Exchange Hosted Services, however, only has value when it's used together with an on-premises Exchange Server, a style that Microsoft sometimes calls an *attached* service.

Similar scenarios can apply to packaged applications. For example, as Figure 3 shows, an organization that uses Microsoft Dynamics CRM can choose to run it as on-premises software, licensing and supporting this software itself. It can also choose to use Dynamics CRM as a partner-hosted service, an offering available from a number of Microsoft partners. Finally, Microsoft itself provides this service as an option called *CRM Live*, running the software in its own data centers. The same software is used in all three cases, making it easier for customers to move from one option to another if their requirements change.

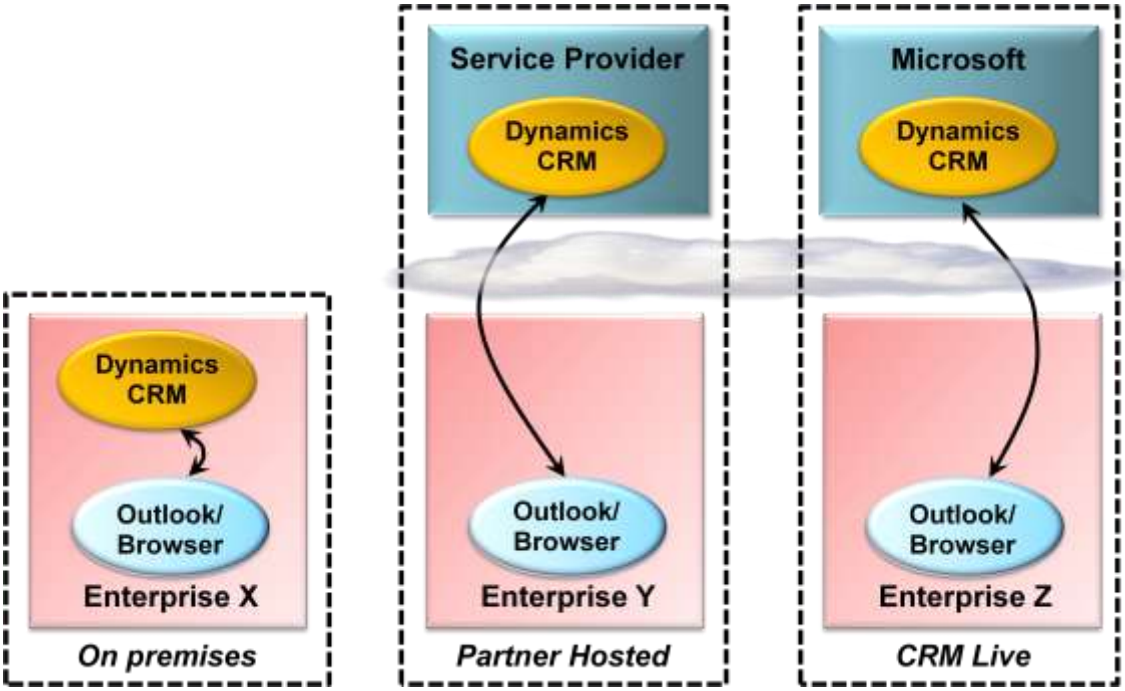

Figure 3: Microsoft Dynamics CRM can be used as on-premises software, as a partner-hosted service, or via the Microsoft-hosted CRM Live

It's likely that the use of on-premises software combined with services will grow. Exchange Hosted Services provides one example of this, but there are plenty of others. For instance, a developer debugging an application with Visual Studio 2008 can move directly into the .NET Framework's source code, with this code downloaded on demand from a Microsoft-provided service. Similarly, the ability for an application to combine, say, geographic data provided by a service such as Microsoft Virtual Earth with its own on-premises data and logic can allow creating better experiences for users. In fact, it's clear that Microsoft plans to move in this direction. According to Ray Ozzie, the company's CTO, "Every one of our software offerings is either a socket for a new attached service that connects to that software offering, or an upgrade or up-sell opportunity to extend a product's value proposition up to the Web".

S+S AND APPLICATION PLATFORMS

Whether an application is used as on-premises software or offered as a service, it must run on some platform. This is a simple idea, but it's important: Even services are really just applications. Figure 4 illustrates this reality.

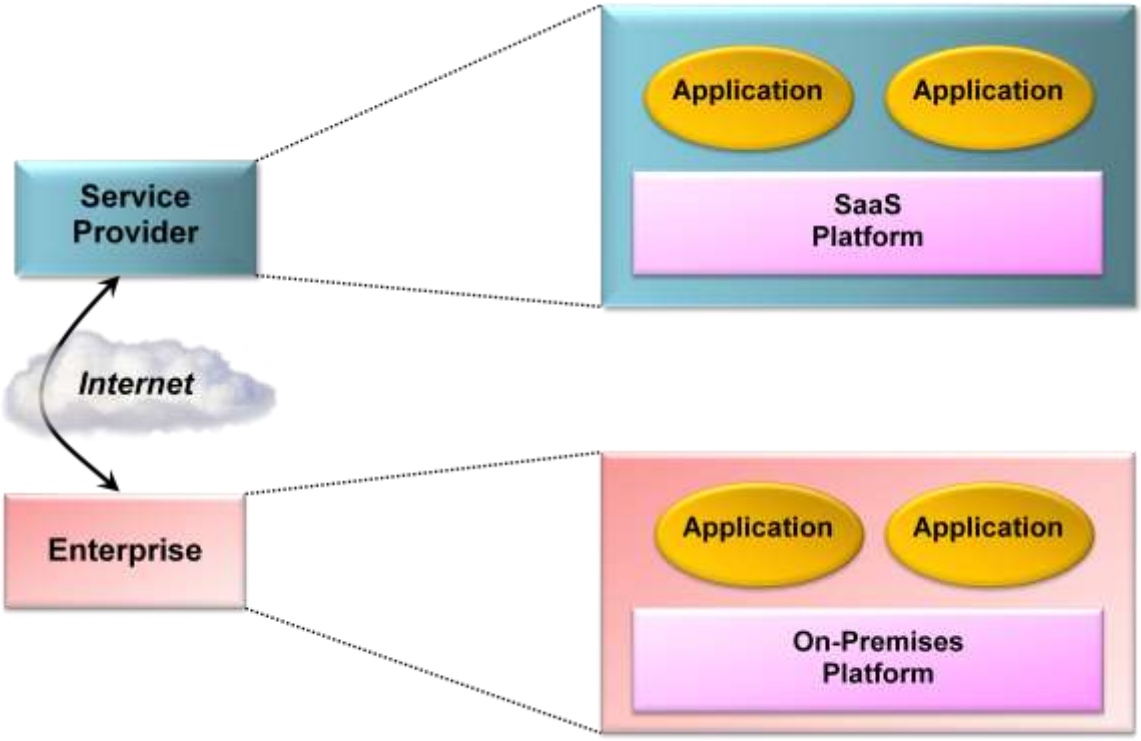

Figure 4: Applications run on platforms, both within enterprises and at service providers

As the figure shows, enterprises run applications on *on-premises* platforms. The majority of new enterprise applications today are built on one of two on-premises platform technologies: the .NET Framework or products based on Java Enterprise Edition (Java EE). Both offer a mature environment with tools, good documentation, and lots of knowledgeable developers.

This kind of stability and experience doesn't exist with *SaaS platform*s. Even the name isn't standard; other terms for the same thing include *Internet* platform, *on-demand* platform, and *cloud* platform. This confusion reflects the fact that thinking of service providers as offering a general platform for applications is a relatively new idea. Yet creating a broadly usable foundation for building Internet-accessible services (that is, for applications running at a service provider) has a great deal to offer. Rather than expanding its own computing resources, for example, an enterprise might instead run some applications at a service provider. Much as utilities provide electric power, computing resources such as processing power and storage can be provided as a utility.

As described later in this overview, on-premises platforms sometimes use different technologies from SaaS platforms. Because of this, the creator of an application often must decide up front whether that software will be deployed within an organization or at a service provider. And because today's SaaS platforms tend to be tied to a specific service provider, moving a SaaS application to another provider might require extensive rewriting. Similarly, an application created for one platform—SaaS or on-premises—might be difficult to move to the other world.

There's no obvious reason why these two kinds of platforms should be significantly different, however. In fact, it's easy to imagine that the technologies they use will one day converge. Having a single platform for both the on-premises and SaaS environments would allow organizations to move applications as needed.

As described later, Microsoft plans to do exactly this, providing a single platform technology for both on-premises and SaaS applications.

SOFTWARE OR SERVICES? WEIGHING THE OPTIONS

Both on-premises software and SaaS have advantages and disadvantages. Using each option well requires understanding the benefits and drawbacks of both. Since every enterprise has experience working with on-premises software, the clearest way to look at this issue is to examine the benefits and drawbacks of SaaS.

For a typical enterprise, the benefits that SaaS can offer include the following:

- Lower costs. If a service provider offers the same services to many organizations, it can take advantage of economies of scale, and so potentially pass those savings on to their customers. Furthermore, with subscription-based services, costs are proportional to usage, and so an organization can pay for only what it uses. Saving money isn't guaranteed, however, since long-run costs are harder to predict. Buying software as a service for a long period might end up costing more than licensing and running equivalent functionality as on-premises software.

- Faster deployment. Rather than wait for the IT department to buy, install, and configure a packaged application—or worse yet, build a custom application—a business decision maker might be able to find a service provider that offers the same solution. (The decision maker can perhaps even avoid dealing with her IT organization altogether, an appealing prospect in some quarters.) The solution can then start providing business value much quicker than it otherwise might.

- Less financial risk: A subscription-based SaaS solution lets an organization avoid the risk of a large up-front investment, since there's no need to buy or build its own software. Some SaaS offerings even provide a try-before-you-buy option, lowering the risk even more.

- Higher reliability: Whether using subscription-based services or more traditional hosted services, a service provider is a specialist in operating this software. Rather than trying to build and maintain a competent in-house staff, using service-based solutions can let an organization benefit from the reliability that service providers offer. Experiences vary, of course; organizations that run their own very reliable data centers might find that even the best service providers don't offer real improvement in this area. Still, relying on a more specialized provider can improve life for some.

- More frequent upgrades: Rather than waiting several years for a new version, then suffering through the pain of migration, a customer can take advantage of new capabilities as soon as they're added to the software. While this places a bigger burden on the SaaS vendor—a buggy upgrade will immediately upset many customers—it can give users a better experience. Still, some customers want to control when upgrades are made to the software they use. In this case, having a SaaS vendor silently change the functionality of an application might be seen as a bad thing.

The benefits of services are clear, and for many organizations, they can be significant. Yet SaaS isn't right for every situation. Along with its benefits, this approach brings some obvious challenges. Among the most important of these are the following:

- Trust: One way to view SaaS is as a form of outsourcing. Putting an important part of an organization's business in the hands of a service provider requires trusting that provider. What happens if it abruptly goes out of business? What kind of disaster recovery procedures does the provider have in place? Especially for mission-critical applications, enterprises need to make sure that their trust isn't misplaced. Service-level agreements (SLAs) are essential, but they're not enough. In the end, anyone who bets on a service provider must have faith in that company.

- Data: It's common for service providers to store data owned by their customers. Having a clear understanding of who owns and can access that data is fundamentally important. Once again, trust is paramount. Imagine the pain caused by a service provider who refuses to return your data when you wish to leave for a competitor, for example. Even more mundane challenges, such as moving data between different DBMSs at different providers, can make switching hard.

- Regulatory and compliance issues: Failing to comply with government regulations about document retention, privacy, or other issues can have enormous consequences (potentially including jail time for a firm's CEO). Trusting a service provider to address these issues is no small thing. What if your firm stores credit card numbers or personal medical data at the provider, and the provider lets that data be stolen? Can the provider do what's necessary to ensure compliance with any relevant regulations? How can you prove this to regulators and your own internal auditors?

- Integration: Enterprise applications frequently need to connect with other enterprise applications. Integrating a SaaS application running at a service provider with an on-premises application can be more challenging than, say, connecting two applications that both run in your data center.

- Customization: An organization is typically free to customize on-premises software to meet its unique requirements. This might not be true for services. If a single instance of a SaaS application is shared by many users, for example, the opportunities for customization can be more limited.

- Identity: Like any other application, a SaaS application needs to know who its users are. If the service provider maintains its own accounts, each user will need to log in to the application explicitly. The provider will also need to maintain those accounts, adding new users and removing them when necessary. A better solution is for the service provider to accept identities provided by the enterprise, perhaps by supporting standards such as WS-Federation.

- Management: Most enterprises are accustomed to having a current window into the status of their applications. But what happens when one or more of these applications are running at a service provider? While the provider will itself monitor and manage the application, enterprises that use it are also likely to want to have some insight into the application's status. This means that organizations must either find a way to integrate monitoring of SaaS applications into their current management console, a tall order today, or invent some ad hoc approach to monitoring SaaS applications.

- Supporting users: As with on-premises software, SaaS applications are bound to generate calls to the help desk. Will an enterprise force its users to be aware of the distinction, having them call the provider's help desk directly for questions about a SaaS application? What happens when organizations have multiple SaaS applications, all with their own help desks? And if the enterprise's help desk takes all calls, then interacts itself with provider help desks, what happens if a provider isn't

responsive? An enterprise help desk that fails to meet its own SLAs due to ineffective provider help desks won't be a happy place.

This list of challenges makes clear that even though services are the right solution in many situations, they're not always the best option for enterprises. Exactly how much each of these concerns matters depends on how important the application is. A mission-critical application being hosted by a new service provider would need plenty of solid assurances before it replaced any on-premises equivalent. A non-mission-critical application being hosted by a provider with whom you already have a relationship might raise significantly fewer concerns.

It's worth pointing out that the benefits of SaaS are evident to business decision makers, while many of the challenges lie more in the domain of IT managers. This can lead to conflict, since a service that's very attractive to a business leader might create headaches for her IT department. Coupled with the fact that business units might be able to contract for services without even telling their IT staff, living in an S+S world is likely to result in at least a few tense interactions between these two domains.

EXPLOITING AN S+S WORLD

On-premises software can provide value to every enterprise. Services can also provide value to every enterprise. Making the right decisions in an S+S world requires understanding the trade-offs, then creating the right hybrid for your organization.

Not all on-premises software in an enterprise will transition to services, of course. The situation is analogous to the advent of Internet-based commerce, when zealots argued that Web-based retailing would doom brick-and-mortar stores. The reality is that the two are complementary: Both offer value, both are common today, and many retailers offer both options.

Approaching the S+S world in this same way makes sense. Enterprises can adjust their mix of on-premises software and services as their needs change. While services can offer lower costs and quicker deployment, on-premises software often allows greater customization and more control. Mixing the two appropriately can help organizations function more effectively at lower cost.

For example, moving infrastructure services such as email to a service provider might be an organization's first step in exploiting an S+S world. An enterprise that currently runs its own Exchange Server infrastructure could find that switching to a hosted Exchange service is significantly cheaper, more reliable, or both. Or think about a smaller organization that initially supports its sales force with a service such as Dynamics CRM Live. As it grows, it might find that bringing this service in-house as on-premises software makes more sense. Perhaps its usage is so high that the on-premises approach becomes cheaper, or maybe it needs more customization and control over the software. Also, as mentioned earlier, Microsoft plans to add a services component to most of its on-premises software. As with Exchange Server, where Exchange Hosted Services provides additional functions as a service-based option, expect the possible combinations of software and services to grow.

It's plausible, although by no means certain, that a great deal of what is currently done by on-premises software will move to external services. In manufacturing, once the barriers to outsourcing came down, nearly everything that could be done externally was moved out; work went to where it could be done most effectively and most cheaply. The same thing may happen with IT. Operating and managing applications isn't part of the core business for most enterprises, yet it soaks up money and managerial

time. Why not let someone else do it for less? Keeping only the core IT support for key business processes—the most differentiating parts—as on-premises software while farming out all commodity functions as services may become the norm.

Going forward, every IT decision maker needs to understand the tradeoffs between on-premises software and external services, as well as how they can be used together. Getting the most from each option means choosing the S+S combination that best fits your organization.

A CLOSER LOOK AT SERVICES

The software part of S+S is familiar; it's been part of our lives for decades. The services part is less clear, however, and so it's worth a bit more examination. This section starts with a look at how services can be provided, then walks through some options for pricing services.

PROVIDING SERVICES

Two important aspects of providing services stand out. The first is figuring out who the customer is: Does the service target businesses or consumers? The second is determining how much those customers should share and thus how the application should be built. This section looks at both.

Choosing Customers: Businesses or Consumers?

Internet-accessible services are useful for both businesses and consumers. Yet these two kinds of services have quite different requirements. Reflecting this, Microsoft groups many of its services into two different groups: The Online family of services targets enterprises, while services offered under the Live banner are aimed at consumers and small businesses.

Think about email, for instance. An enterprise relying on Microsoft or another service provider for hosted Exchange will be deeply unhappy if that service is unavailable, especially if the outage is unplanned. The business pays for this service, and it expects to get its money's worth from the provider. Yet a consumer using, say, Windows Live Hotmail, isn't likely to be quite as upset if this email service is unavailable for some reason. Hotmail is a free, ad-supported service, and so it's not reasonable to expect the same level of reliability as in a commercial mail service.

In fact, a business user of any commercial service is likely to have a service level agreement (SLA) in place with its service provider. The SLA will specify exactly what level of service must be offered, probably spelling out penalties for failing to meet this level. While consumer services might have an implied SLA—Microsoft would get bad press if Hotmail weren't reliable, for instance—business services typically need a more formal arrangement.

Services aimed at businesses also have other requirements. As mentioned earlier, they commonly must be able to manage identity, support users, and perform other functions that aren't relevant to consumers. With Google Apps, for example, the Standard edition offers a free, ad-supported service with basic email, calendaring, and document creation. The Premier edition provides the same services but requires a monthly subscription fee—users don't see ads. It also includes an SLA for email, the ability to integrate with existing enterprise applications, a help desk, and other functions necessary for business users.

Business might also put a premium on the ability to change service providers. Being locked into a single provider can be problematic if that provider becomes unreliable, too expensive, or difficult to deal with. One way to encourage this is by having multiple service providers use the same platform, a topic that's addressed in a bit more detail later in this overview.

Choosing an Implementation Style: Single-Tenant or Multi-Tenant?

Service providers can take different approaches to implementing their services. An important differentiator between these approaches is how much is shared across different customers. One way to think about this is to contrast *single-tenant* applications with *multi-tenant* applications.

In a single-tenant scenario, a service provider offers a unique instance of an application for each organization that uses it. This approach is simple, and it can be used with most existing applications. Figure 5 shows how it looks.

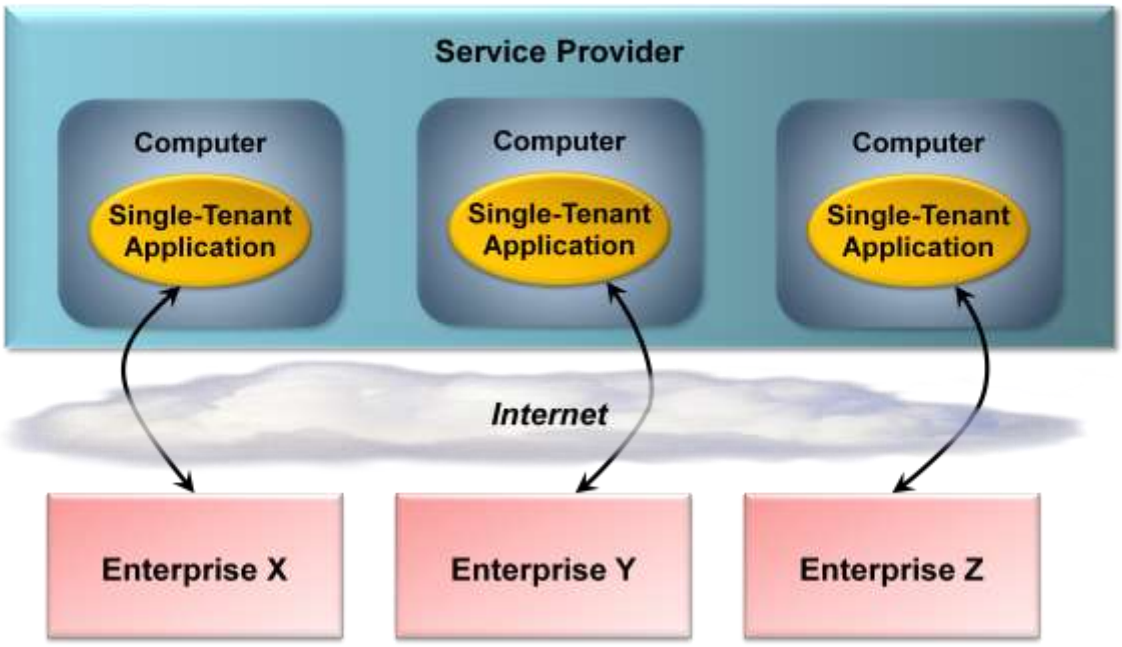

Figure 5: A single-tenant application runs as a separate instance for each customer

Running a separate instance of an application for each customer has a number of benefits. Most important, it provides a significant amount of isolation between these different instances. To see why this is important, think about how hard it is for many businesses to trust their applications and data to a service provider. Suppose this same provider also works with one of a firm's competitors—can the provider be trusted not to leak information? Single-tenant applications can provide a good solution to this concern. Depending on how much isolation a customer needs, the service provider can run each instance of an application in its own virtual machine, its own physical machine, or for large customers, even provide a separate data center. Single-tenant applications also give the provider more freedom to customize each customer's application and environment as required.

The downside of this approach is that it doesn't necessarily allow many economies of scale. Because less is shared across customers, a service provider will likely have fewer opportunities to reduce its costs, and

so need to charge higher prices. Still, many organizations have found that single-tenant applications are the right approach.

The alternative to providing SaaS via a single-tenant application is to let all customers use a single instance of the same application, all relying on the same code base. Unlike a single-tenant application, customers that rely on a multi-tenant application share a great deal. Figure 6 illustrates this idea.

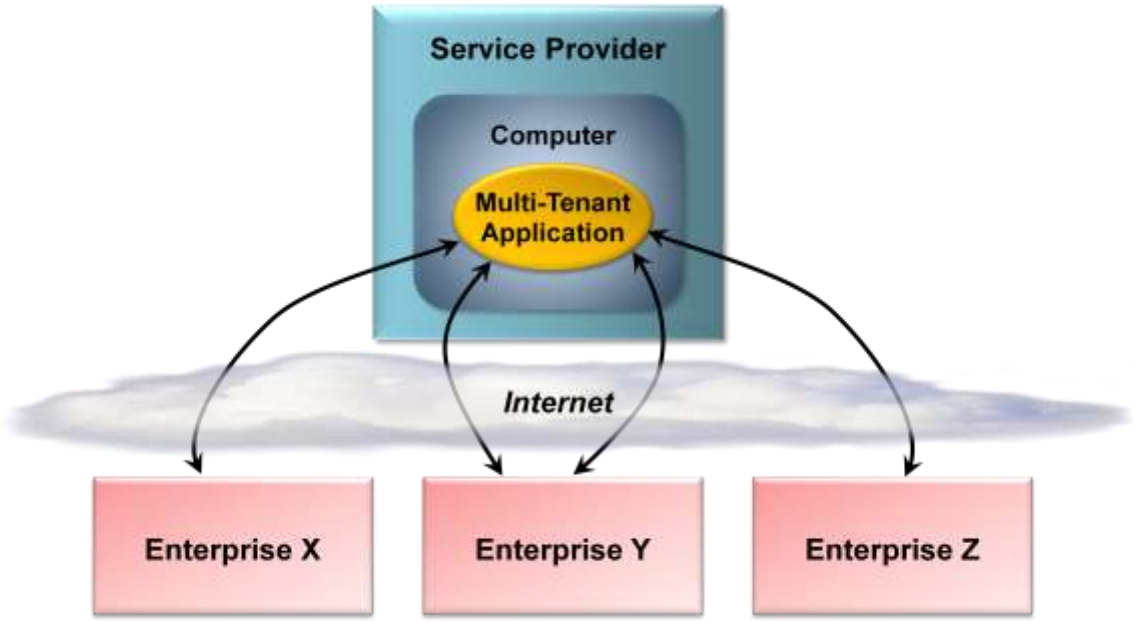

Figure 6: A multi-tenant application lets multiple customers share a single instance

Customers using a multi-tenant application place a significant amount of trust in that application's provider. While each customer can have its own data, it's very possibly sharing both the computer and the application itself with competitors. This kind of sharing also makes customization more challenging, since each customer accesses the same instance of the application. Also, as with any shared application, meeting SLAs can be more difficult for the service provider.

Yet multi-tenant applications definitely have an important role to play. Because they share so much, they can have the lowest possible cost to operate. If those low costs translate into low prices, as they generally will, customers can find this an attractive option.

Multi-tenant applications get lots of attention in the SaaS world, but it's important to understand that they're just one approach. (In fact, true multi-tenant applications aren't especially common today.) The truth is that giving up isolation of applications and data isn't always acceptable to enterprises. Besides, from the customer's point of view, single- and multi-tenant applications can both provide much the same service. Which one an enterprise chooses depends on its need for isolating applications and data, its budget, and other factors.

PRICING SERVICES

Services can be differentiated by who they target and how much they share, as just described. Another important differentiator is how a service is priced. At one extreme, a customer might license the software

itself, then pay a recurring fee to a provider to run and manage that software on the provider's computers. It's debatable whether this really qualifies as SaaS; viewing it as hosting might be more accurate.

When the service provider itself licenses or builds the software, however, then charges its customers a fee to use it, this is clearly software as a service. The fee might be transaction-based, with more usage incurring higher costs, or it might be a fixed price per user with unlimited usage for some period. In any case, some kind of subscription-based pricing is typically an integral part of SaaS.

APPLICATION PLATFORMS IN AN S+S WORLD

Just as the move to an S+S environment changes applications, it also changes the platforms on which those applications run. This section first looks at a specific example of this change, Microsoft's BizTalk offerings, then steps back for a more general perspective on application platforms in this new world.

HOW S+S CHANGES APPLICATION PLATFORMS: THE BIZTALK EXAMPLE

Especially in large organizations, a business process often depends on more than one application. Connecting those applications can help automate the process, making it faster, more consistent, and more reliable. This kind of integration can be done using on-premises software such as Microsoft's BizTalk Server. It can also be done at a service provider, however, as shown by Microsoft's BizTalk Services. Comparing the two provides a good example of how application platforms are changing in an S+S world.

Integration Using On-Premises Software: BizTalk Server

Viewed abstractly, BizTalk Server provides two main functions for integrating applications. Its *messaging* component allows interacting with diverse applications using various communication mechanisms. Its *workflow* component (also referred to as *orchestration*) allows defining business logic that drives the integration process. Figure 7 shows a simple picture of how this looks.

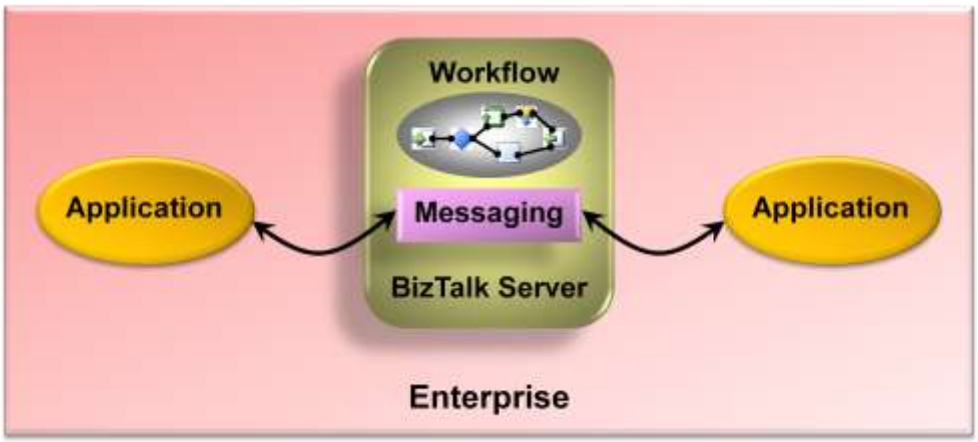

Figure 7: BizTalk Server integrates applications with workflow and messaging running in on-premises software

Because it must connect with other software in many different ways, BizTalk Server messaging is accomplished using *adapters*. Each adapter implements support for some kind of communication, such as Web services, message queuing, or something else. Beginning with BizTalk Server 2006 R2, these adapters can be implemented using Windows Communication Foundation (WCF), Microsoft's general communication technology for .NET Framework applications. And while workflows are currently implemented using a BizTalk-specific orchestration engine, Microsoft has announced that a future release of the product will instead use Windows Workflow Foundation (WF), a standard workflow engine that's also part of the .NET Framework.

As with applications, it sometimes makes sense to license and run an integration product such as BizTalk Server as on-premises software. Yet it can also make sense to use integration services offered by a service provider. As described next, this is exactly what BizTalk Services allows.

Integration Using Services: BizTalk Services

The fundamental problems of integration remain the same whether it's done on-premises or via a service provider. Diverse applications still need to communicate, and business logic must drive the process. Yet a few changes are in order when a service provider does these things, as Figure 8 suggests.

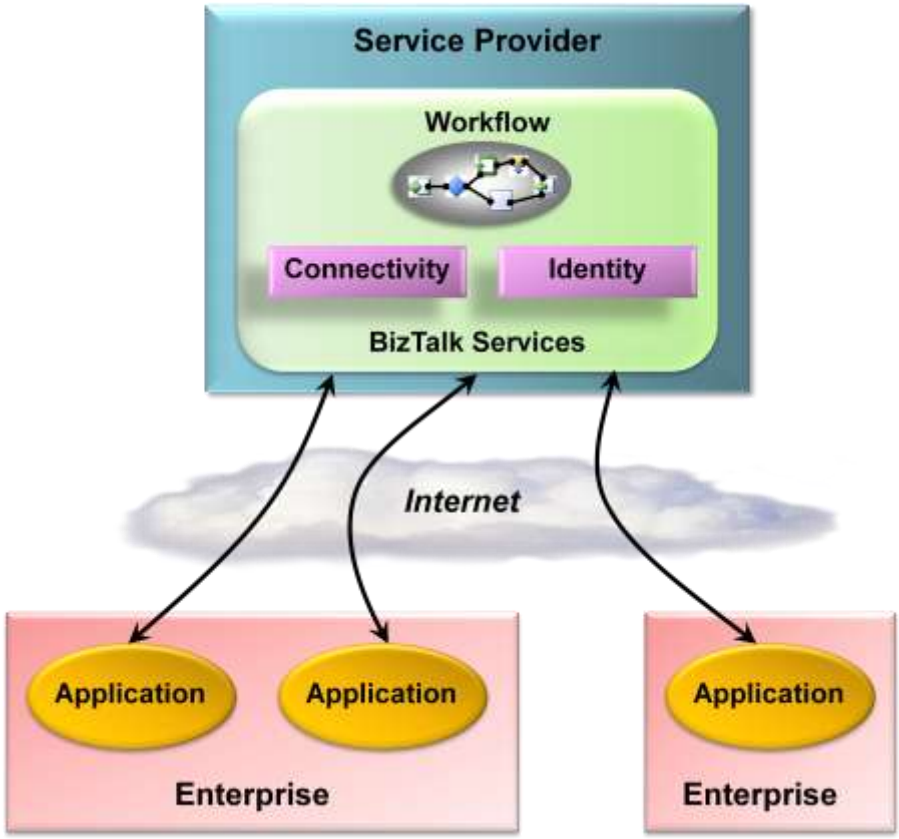

Figure 8: BizTalk Services integrates applications with workflow, connectivity, and identity services running at a service provider

As the figure shows, making integration services Internet-accessible allows connecting applications both in the same enterprise and across enterprises. Wherever the applications run, BizTalk Services provides a workflow engine for executing business logic to control the integration process. Like the future version of BizTalk Server, this engine is implemented using WF.

BizTalk Services also provides communication between applications through its *connectivity* service. Built using WCF, this service can communicate with applications that use WCF or other technologies. Yet while it's conceptually similar to BizTalk Server's messaging, the connectivity service isn't quite the same. It's designed to help applications communicate through firewalls, for example, a problem that's very real with service-based integration. It also allows assigning a globally addressable name to an application-provided service (which lets that service be located from anywhere on the Internet), message multi-casting, and more.

To manage the identities of users, BizTalk Server can rely on the identity infrastructure already available in an enterprise, such as Active Directory Domain Services. BizTalk Services doesn't have this luxury—it's an Internet-accessible service—and so it must provide its own identity solution. To do this, BizTalk Services implements a security token service (STS). Accessible via the standard protocol defined by WS-Trust, this identity service issues tokens that allow external applications to access the connectivity service.

Reflecting its supporting role, Microsoft sometimes describes BizTalk Services as an example of a *building-block* service. While this technology certainly doesn't offer everything that's available in its on-premises sibling, the basics of integration are there. It's still early days, but BizTalk Services illustrates both what's possible and the kinds of changes that take place when components of the application platform are re-imagined as services. In fact, just as it's possible to view BizTalk Server as an enterprise service bus (ESB), Microsoft sometimes describes BizTalk Services as an *Internet service bus (ISB)*. And although it's not shown in Figure 8, it's certainly possible to use BizTalk Services and BizTalk Server together, combining the abilities of on-premises and service-based integration. For example, an organization might use BizTalk Services to provide a relatively simple way to connect with its suppliers, then use BizTalk Server internally to connect its own applications. Especially for smaller organizations, BizTalk Services might provide lower-overhead communication with partners than more traditional solutions such as electronic data interchange (EDI).

Today, BizTalk Services is hosted by Microsoft. The company's stated goal, however, is to make this software available to other service providers as well. Since it's built on a standard foundation, including WF and WCF, BizTalk Services doesn't require a special platform to run. Wherever it's hosted, providing integration via services allows new capabilities to be deployed more quickly, since there's only one copy of the software to update and test. As both BizTalk Server and BizTalk Services advance, expect to see these two technologies share more code and more functionality. While they're unlikely to be exactly the same—service-based integration will always be somewhat different from on-premises integration—converging the two as much as possible makes sense.

EXAMINING SAAS PLATFORMS

The rise of S+S implies the rise of SaaS platforms. Accordingly, it's worth looking in more detail at this emerging technology. This section describes some of the issues raised by this new style of application platform.

It's useful to distinguish between a true SaaS platform and a programmable service. While both are important, SaaS platforms are likely to have a bigger impact on how and where enterprises run their software. As Figure 9 shows, a SaaS platform lets a service provider run custom applications created by its customers.

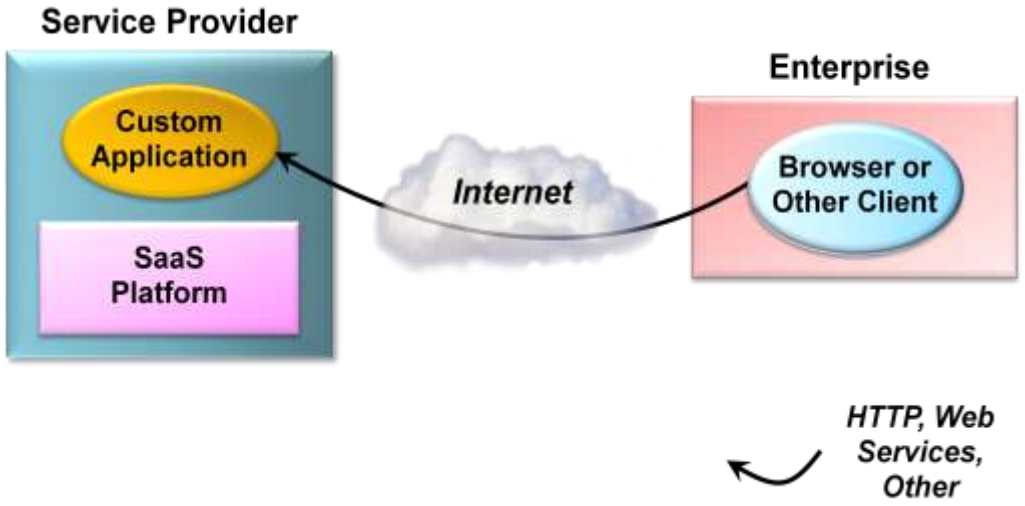

Figure 9: A SaaS platform runs custom applications at the service provider

As the figure suggests, a SaaS platform is analogous to today's on-premises platforms. Like those more familiar technologies, it supports running arbitrary applications. Clients can include browsers or other kinds of software, and they might communicate with the application using HTTP, Web services, or something else. The computing resources used by the application, such as processing cycles and storage, are made available by the service provider.

Several providers offer this kind of platform today. While there's plenty of variation, it's useful to divide these offerings into two broad categories:

- Platforms that provide a general computing resource. A visible example of this is Amazon's Elastic Compute Cloud (EC2) and Simple Storage Service (S3). EC2 provides customer-specific virtual machines (VMs), while S3 provides storage for applications running in those VMs and for other purposes. Customers can upload their own applications, then pay only for the resources those applications consume.

- Platforms that provide higher-level services focused on supporting applications. One example of this is Force.com, Salesforce.com's platform for creating data-oriented business applications. Along with support for multi-tenant applications, this platform also includes its own programming language, Apex. Another example is the platform provided by Microsoft's Dynamics CRM. Also focused on supporting data-oriented business applications, it lets developers create business logic using standard .NET languages and technologies. Because the same code is used whether the platform is deployed on-premises or in a service provider, applications built on this platform can run in either place.

Organizations such as Amazon, Salesforce.com, and Microsoft have invested a significant amount in creating scalable infrastructures for their own applications. It's not surprising that they've chosen to make those infrastructures available to the outside world as SaaS platforms. This style of application platform can have a great deal to offer for both independent software vendors (ISVs) and enterprises.

SaaS platforms support custom applications created by customers. A programmable service is quite different. While it is in some sense a platform—it allows custom applications to be built that use it—those applications don't run at the service provider. Instead, as Figure 10 illustrates, they run on some on-premises platform in an enterprise or elsewhere.

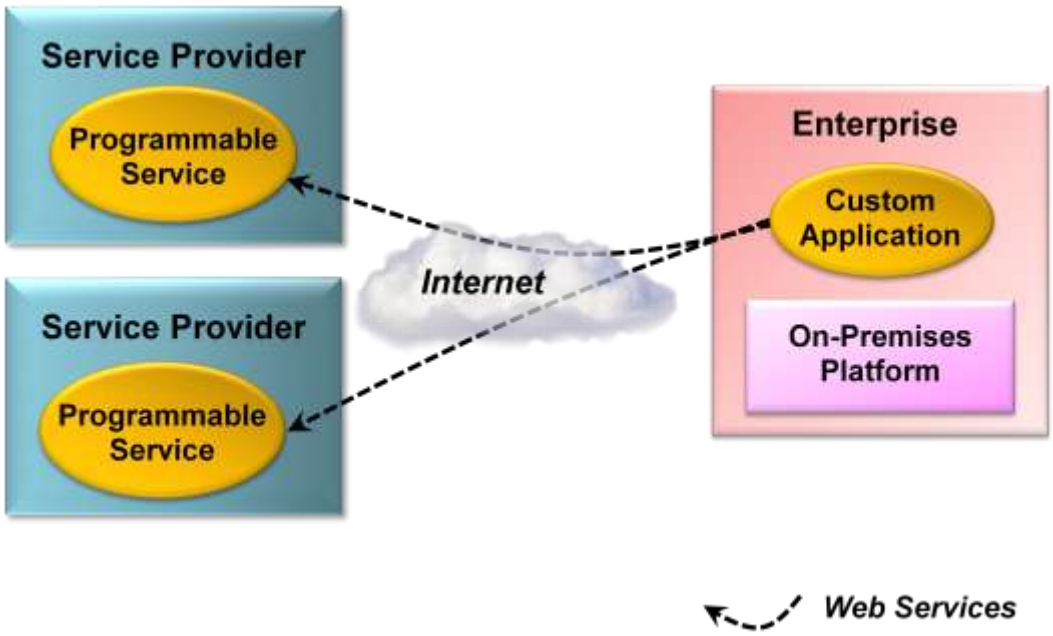

Figure 10: A programmable service exposes Web services to custom applications running on an on-premises platform

As the figure shows, this approach supports only the programmable services—that is, the applications—offered by a particular service provider. A custom application accesses the programmable service using Web services. Many Internet sites provide this today, including Windows Live Services, Flickr, and others. A common use of this kind of service is to create mash-ups, combining various services with custom logic to create an application.

It's also possible for a SaaS platform to be usable as a programmable service, exposing various Web services. For example, Amazon's EC2 and S3 expose Web services that let clients upload and run code, access data, and more. Similarly, Salesforce.com allows its CRM application to be customized either by a remote application that calls exposed Web services, or with Apex code running at the service provider. Yet while both programmable services and SaaS platforms are useful, only SaaS platforms have the potential to change where ISVs and enterprises create and run applications. The next section explains why this is so.

SaaS Platforms for Custom Applications

Think of the challenges an ISV faces today. Once an application has been created for some on-premises platform, it must be tested in all of the environments in which it will run, perhaps on multiple operating systems. The application must then be installed on machines at each customer. Updating it requires changing this installed code at each customer's site, maybe even multiple copies of that code. And if customers don't provide enough computing power for the application, the ISV is likely to be blamed for creating a slow application.

Now think of an ISV's life if it instead builds an application for a SaaS platform. The application will run in only one environment, which reduces testing costs. Once it's ready, the application is immediately available to any customer with Internet access anywhere in the world—no installation at customer sites is required (at least for browser-accessible applications). Updating the application requires changing just one copy of the code. And the service provider is responsible for coming up with enough computing resources to meet customer requirements.

This is an appealing approach, one embraced by a number of SaaS ISVs today. Yet once stable, reliable, and well-priced SaaS platforms are available, many more ISVs are likely to find this model attractive. Enterprises are beginning to confront a world where major ISVs offer two versions of an application: a more expensive but more customizable package installed on-premises, along with a cheaper option offering a more generic service running at a service provider.

Enterprise developers who create custom applications will also see benefits from the rise of SaaS platforms. Some of these benefits are identical to those described earlier for SaaS, such as letting a service provider run the application and perhaps getting a more reliable infrastructure to run it on. Enterprises can also exploit what will likely be large, cheap computing and storage resources available from service providers. For example, an enterprise that must cope with an annual spike in demand, such as a Christmas rush, could run one or more of its applications at a service provider. Rather than investing in permanent data center upgrades, it could instead rely on this expandable computing resource, paying only for what it uses.

Still, running enterprise applications at a service provider will have some downside. Most of the challenges of SaaS described earlier, such as meeting regulatory requirements, integration hurdles, handling identity, and effective management, also apply here. The issue of trust once again comes to the fore, especially for mission-critical applications.

Another challenge is choosing the right SaaS platform. Creating an effective SaaS platform is no easy thing. Just as with an on-premises platform, it must provide a secure runtime environment together with some kind of storage mechanism. It should also provide built-in scalability, and especially if it supports multi-tenant applications, a way to limit the damage that an errant application can do. For applications that must charge their customers, such as those built by ISVs, the platform should offer standard ways to meter usage and provide billing services. And like any other platform, a SaaS platform requires tools and documentation, ideally in multiple languages. In other words, SaaS platforms must address all of the challenges of on-premises platforms and more.

For both enterprises and ISVs, building an application that runs on a SaaS platform means first deciding which platform to bet on. As described earlier, SaaS platforms can take many forms. A provider might

simply offer standard Windows Server System software, for example, with little effort to add extra value. Another alternative is to provide virtual machines, as Amazon's EC2 does.

One more option is to build on a SaaS platform that provides higher-level application services, such as those provided by Salesforce.com and Microsoft Dynamics CRM. In some cases, this is clearly the right choice. Extending an existing Salesforce.com application using Apex can make sense, for example. Yet building a new application on a provider-specific SaaS platform locks the application into that provider. If the relationship with the service provider goes bad, moving the application to another one can be difficult. Like choosing an on-premises platform, selecting a SaaS platform is an important decision.

LOOKING AHEAD: MICROSOFT'S COMMON PLATFORM FOR ON-PREMISES SOFTWARE AND SAAS

Today, the application platforms used within an enterprise and at a service provider can look quite different. But why should this be true? Is there any compelling reason why an on-premises platform and a SaaS platform can't look much the same to developers? As the example of BizTalk Server and BizTalk Services shows, there are differences but also large overlaps in the services required for both.

Microsoft's stated direction with its Oslo investments reflects this reality. Their goal is to create a general-purpose SaaS platform that lets developers use the same tools and the same fundamental skill sets as with their on-premises platform. By providing similar APIs in both platforms, applications can potentially be created without deciding up-front whether they'll run on-premises or as a service. An application might first run locally, then be moved to a service provider for more capacity or lower cost. A multi-tier application might even run partly on-premises and partly at a service provider. Think, for example, of an application that requires significant computing resources but works on data that for regulatory reasons must remain within the enterprise. Perhaps the application could run its business logic at a service provider, taking advantage of cheap computing power, while accessing locally held data over a fast and secure connection. This combination of both environments is a good example of what can be possible in an S+S world.

Microsoft's announced plan is to make this new software available both in its own data centers and at other service providers who license the platform. Because the same SaaS platform will be available at multiple providers, customers will face less risk of being locked into a single service provider. The goal is to make the worlds of software and services as similar as possible, making life easier for anyone who works in both.

CONCLUSION

Enterprises already use both on-premises software and services, but there's still plenty of change to come. For most organizations, the primary impact of S+S will likely be moving more and more capabilities— infrastructure, packaged applications, and eventually even custom applications—from computers within the enterprise to computers at a service provider. The economics of services will be too strong to resist.

The result will be a shift in the software landscape that affects software vendors as well as users. One way to think about this new world is by picturing a simple two-dimensional grid, with various vendor offerings positioned on this grid. Figure 11 shows how this looks, with representative examples in each quadrant.

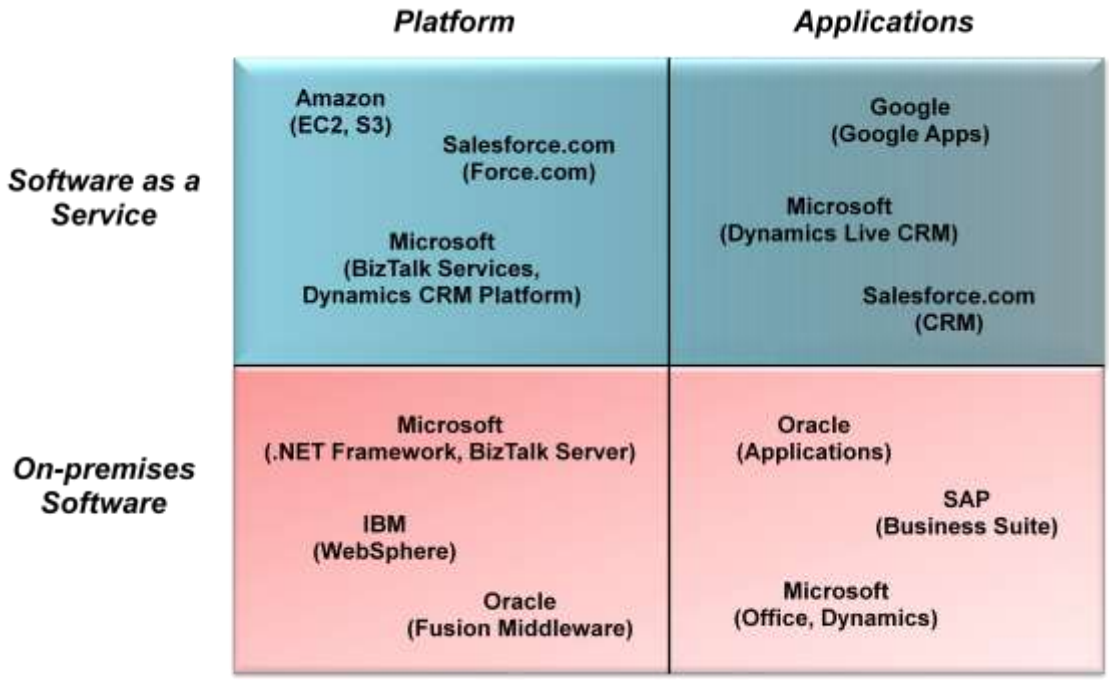

Figure 11: As these examples show, offerings from enterprise software vendors can be divided into four quadrants

The vertical axis indicates whether an offering is on-premises software or SaaS, while the horizontal axis distinguishes between platforms and applications. The lower left quadrant, for example, shows familiar vendor offerings for on-premises application platforms, such as Microsoft's .NET Framework, IBM's WebSphere, and Oracle's Fusion Middleware. The lower right quadrant contains examples of on-premises applications such as Oracle Applications, SAP BusinessSuite, Microsoft Office, and Microsoft Dynamics.

The top row in this diagram contains more recent offerings that are less common in today's enterprises. The upper right quadrant shows representative examples of SaaS applications, including Google Apps, Microsoft Dynamics Live CRM, and Salesforce.com. The upper left quadrant contains some early examples of SaaS platforms, such as Amazon's EC2 and S3, Salesforce.com's Force.com, and Microsoft's BizTalk Services and Dynamics CRM platform.

The goal of this diagram is not to provide a complete listing of all products available today. Rather, the figure offers a useful way to think about the S+S world using representative examples. It's also worth noting that Microsoft is the only vendor appearing in all four quadrants. This shouldn't be surprising; Microsoft is the world's biggest software company, which gives it both the means and the motivation to pursue all four areas.

While the use of services is bound to grow, on-premises software will continue to have significant value in an S+S world. Getting the balance right between software and services will take some work, and it won't be a fixed ratio for most organizations. Instead, decision makers will need to balance the trade-offs between the two as technology and business requirements change. While a typical enterprise today uses more on-premises software than services, this might not be true a decade from now. In fact, it's easy to imagine a world where the default is to use a service, with on-premises software deployed only when absolutely necessary. While SaaS is too immature to make this a reality today, it's not too soon for

organizations to explore the potential benefits of this approach. Once a software + services perspective becomes the norm, we may one day marvel that we didn't always do things this way.

ABOUT THE AUTHOR

David Chappell is Principal of Chappell & Associates (www.davidchappell.com) in San Francisco, California. Through his speaking, writing, and consulting, he helps information technology professionals around the world understand, use, and make better decisions about enterprise software.