Developing Web services using the Microsoft.Net platform: technical and business challenges

Wendy L. Currie
Xinkun Wang and
Vishanth Weerakkody

The authors

Wendy L. Currie, **Xinkun Wang** and **Vishanth Weerakkody** are all based at the Centre for Strategic Information Systems, Brunel University, Uxbridge, Middlesex, UK.

Keywords

Computer networks, Information services, Internet, Electronic commerce

Abstract

This paper discusses an ongoing research programme, which explores the development of the software-as-a-service business model by different service providers (xSPs). With the demise of the first phase of the ASP market, due to the failure of vendors to provide business value to potential and existing customers, Web services promise to resolve some of these problems by integrating software applications across heterogeneous technology platforms and business environments. Whether this will be achieved is the subject of continuing debate. This paper presents the preliminary findings from a study, which uses the Microsoft .NET technology platform to develop Web services. Two Web-enabled prototype databases are discussed. The paper concludes that Web services is still relatively new, but if key technical and business challenges are resolved, it may provide value for the customer where ASPs failed.

Electronic access

The Emerald Research Register for this journal is available at
www.emeraldinsight.com/researchregister

The current issue and full text archive of this journal is available at
www.emeraldinsight.com/1741-0398.htm

The Journal of Enterprise Information Management
Volume 17 · Number 5 · 2004 · pp. 335–350
 · ISSN 1741-0398
DOI 10.1108/17410390410560973

Introduction

The economic downturn of the last few years has witnessed the demise of thousands of dot.com or Internet businesses worldwide (Hagel, 2003). Despite the promises of value creation from e-business, the digital economy, the networked organisation, e-business models have been poorly designed and executed. The hype surrounding the application service provider (ASP) industry is a good example of this, as vendors tried to convince small and medium businesses (SMBs) of the benefits in moving to a remote, hosted delivery model (Kern *et al.*, 200X). SMBs, however, remained unconvinced about the value of using ASPs, especially as procuring commodity applications (e-mail) on a price per seat/per month basis offered few business benefits (Ekanayaka *et al.*, 2002). But using ASPs for more complex software applications hosting exposed the customer to additional risks, such as data security infringement and theft (Currie, 2003). The first phase of the ASP market was, therefore, characterized by flawed and failed e-business models, with few paying customers, and the demise of numerous ASPs.

Recognizing the need to add value to the customer, the second phase of the ASP market must address important issues from resolving data security problems to offering integrated and interoperable e-business solutions. Web services offer a new paradigm for connecting business processes independent of their underlying implementation. Unlike, the first phase of the ASP market, Web services are designed to extend the life of legacy systems. They represent a new approach to the perennial problem of integrating disparate applications within or between organisations. In the past, consistent implementation was necessary to enable distributed application interactions. But with Web services, the playing field is levelled, enabling the construction of new composite applications using business functions from other internal or external applications without being dependent on an implementation. This services-orientated approach aims to integrate applications within an organisation. Web services do not rely on an extensive and scalable infrastructure. As Web services become more complex, development scale will remain low, suggesting a more evolutionary approach based upon continuous improvement, rather than revolutionary change. The key difference between a traditional Web application and a Web service is that the information that is returned can be easily integrated into an intranet, Internet, server or desktop applications. Web services should be

The Journal of Enterprise Information Management
Volume 17 · Number 5 · 2004 · 335–350

thought of as applications that have been written that are accessed by other applications rather than by people directly, effectively a Web site for a computer to use.

The lessons learned from the dot.com shakeout show that service providers[1] (xSPs) must focus upon value creation for the customer to become medium or long-term survivors in this evolving ecosystem (ref). xSPs need compelling services to sell – and reliable infrastructure platforms on which to host those services. They need to monitor and measure service use and the quality of the services they deliver. They must offer attractive pricing terms, be able to bill their subscribers accurately and efficiently, and to adjust pricing flexibility to meet changing market conditions (ref). No single provider can offer a full service to customers alone (ref). xSPs need to draw on partners – including platform vendors, application developers, complementary xSPs and others, to provide a seamless software-as-service customer offering. xSPs that develop solutions need to pay particular attention to development tools, and the types of infrastructures and platforms on which they will run these offerings.

In recent years, software and platform vendors have brought Web-optimized applications and new generations of scalable, reliable middleware platforms to market. The rise of XML-based Web services potentially offers xSPs more flexibility to develop differentiated services to generate new revenue streams. If the promises of Web services are fully realised, the software as a services model could reach its full potential (ref).

This paper tracks the evolution towards the software-as-a-service model by examining the business and technical drivers for Web services, with particular attention to how they will resolve the problems, which beset the first phase of the ASP industry. It discusses a funded research programme, which aims to develop two prototype databases using the Microsoft™ .NET technology platform. These databases are intended to help customers and vendors (xSPs) evaluate the market offerings, benefits and risks from the software-as-a-service business model.

The paper is divided into four parts. Part one gives an overview of the definitions of Web services and their evolution as a service provider, which will integrate disparate software applications across heterogeneous organisations. Part two gives some brief examples of how Web services will enhance the software-as-a-service model, by drawing from case study research on specific software applications. The case of Amazon.com will be discussed. Amazon's Web Service efforts are a result of its popular associates program. The members of the associates program share the 15 per cent profit from the purchasing of customers they introduce to the Amazon On-line store. Instead of providing simple links referencing to the Amazon Web site, the new Amazon Web Service facility exports all business functionalities through Web services technology to implement its online shopping function on its associates Web site. The Amazon Web Service case study provides a useful example of an existing Web services application.

Part three introduces the research programme, which evaluates e-business models (ASP and Web services) from xSPs. The research programme combines the technical development of two prototype databases with qualitative research intended to match vendor offerings against customer requirements. In this paper, we focus upon the technical aspects of Web services. Part four presents the development of the two prototype databases. The first is a customer evaluation tool designed to enable SMBs to evaluate the benefits and risks of the ASP business model in five categories: delivery and enablement; integration; management and operations; client/vendor partnerships; and business transformation. This database comprises key performance indicators (KPIs) in each of these categories, and respondents are asked to assess their importance to their own business requirements on a 1-4 scale (1 = not important). The second is an xSP market segmentation database, which delineates xSPs in a number of categories, including horizontal, vertical and technology platform providers. For example, a horizontal service provider may offer generic business software (i.e. HR, accounting, travel and expenses, supply chain management). A vertical service provider will target specific industry sectors/segments (i.e. health, travel, IT, manufacturing/logistics). A technology platform service provider is likely to partner with horizontal and vertical firms to offer IT infrastructure (i.e. data centre, networking, telecoms, co-location services). By adopting Microsoft's .NET platform, the research project provides a backend data retrieval and analysis application to users via SOAP/XML protocols.

The evolution of Web services

Web services are emerging as a systematic and extensible framework for application-to-application interaction. Built upon existing Web protocols and open XML standards (Curbera *et al.*, 2002) information can be seamlessly passed between remote applications running in different environments. This architecture heralds the way for Web services, promising to provide real

The Journal of Enterprise Information Management
Volume 17 · Number 5 · 2004 · 335–350

business delivery mechanisms, system integration and strategic opportunities to organisations. This new distributed computing model has been described as the next-generation of service-oriented Internet applications (Clark, 2002). In order to contribute to the development of Web services, comparing the Web services architecture and its precursors both from a technical and business computing perspective. Definitions of Web services vary, with some pundits focusing on the technical features and others, on the business benefits. The following capture some popular definitions of Web services.

> A Web Service is a software application that is identified by a URI whose interfaces and bindings are capable of being defined, described, and discovered by XML artefacts, and supports direct interactions with other software application using XML-based messages via Internet-based protocols. (W3C Web services Architecture group, www.w3.org/TR/wsa-reqs)

> A collection of business functions or capabilities taken from a single or multiple software applications that, when bundled together, can be published to a network using standard XML-based protocols for use by other applications. Each Web Service is a bundling block that enables the sharing of software functionality with other applications residing outside of the Web Service's native IT environment. (Triple Tree)

> Web services are loosely coupled software components delivered over Internet standard technologies. A Web Service represents a business function or business service and can be accessed by another application...over public networks using generally available protocols. (Gartner Group)

> The promise and hype of Web services is that...applications will begin to function as services that can be identified, located, accessed, compiled, and assembled in pre-specified configurations, dynamically, in real time. (IDC)

> Discrete software components that run on top of the Internet as if it – along with the diverse systems it connects – constituted a huge distributed operating system...Web services are analogous to software components such as Enterprise Java Beans, which make up the elements of a full application that runs on a single-server operating system. (Summit Strategies)

> Web services are a set of standards and protocols, embraced by all major technology providers, that allow software resources to be shared, combined, used an re-used within and between organisations. (Jyoti Banerjee, CEO, MyBusiness.net)

The drive towards Web services is built upon the premise that significant benefits will arise from integrating software applications across heterogeneous organisations, sites and departments. According to Gartner Group,

> The need for e-commerce, collaborative commerce (c-commerce) and involvement with other divisions or departments is pushing vendors to honor enterprises' requests for easy integration, if not out-of-the-box then certainly without having to rely on extraordinary resources....Through 2006, service-orientated development will change the way software is built, packaged and sold by more than 80 per cent of ISVs. By 2005, new licences for software that use Web services standards will represent £21 billion in sales (Gartner, 13 November 2002, p. 2).

Web services pose both a technical and a business challenge to customers and vendors. As we saw with the first phase of ASPs, the difficulty of offering software-as-a-service using applications built in a client server environment was technically challenging. But this was not ameliorated by vendor marketing messages, which failed to convince potential customers of the value proposition of adopting a remote delivery model. Against this background, the next two sections explore some of the technical and business challenges for Web services, which must be addressed if vendors are to create business value for the customer.

Web services: the technical challenge

Rather than being developed in a vacuum, divorced from any practical application, Web services must be seen as a natural evolution to the commercial application of the Internet. The Internet has evolved from other computing paradigms, integrating the approaches and standards that can be seen to exist for electronic data interchange, enterprise application integration (EAI), distributed object computing and of course the Internet protocol (Aoyama *et al.*, 2002).

From object orientated component computing to Web services

Technically, Web services have been developed from object orientated (OO) and component computing, inheriting several advantages from both technologies. These can be seen as shorter development time, more reliable code and more efficient use of development resources. Both OO computing and component computing cannot solve the interoperability issue among heterogeneous Information Technology systems (Orfali *et al.*, 1999). For OO computing, the classical objects only live within a single program, which impedes the communication between the "objects" implemented in other programmed applications (Lim, 1996). For component computing, the plug-and-play middleware capability has never been guaranteed. For instance, EAI implemented in component

The Journal of Enterprise Information Management
Volume 17 · Number 5 · 2004 · 335–350

computing cannot provide a common solution because it attempts to solve the problem using an incomplete set of proprietary technologies. When multiple companies integrate systems that are themselves integrated using different EAI products, developers face the recursive problem of integrating "integration" solutions. Therefore, it is necessary to establish holistic, commonly accepted and standardised Web services, instead of proprietary solutions (Samtani and Sadhwani, 2002a, b; Stal, 2002).

From EAI to ASP and Web services

Enterprise information systems have evolved over the last 30 years. While EAI has been successful in point-to-point integration, it has frequently failed because of complexity and cost (Samtani and Sadhwani, 2002a, b). EAI heavily relies on proprietary technology, which is not only expensive to acquire, operate and maintain, but is complex to implement. As an integrated solution for individual application, EAI creates vendor dependence, or "lock-in", where it is difficult to switch service providers. EAI focuses on data level integration and communication, rather than the application level. For a particular enterprise information system, the customised EAI is quite efficient, but it never creates generic, flexible and reusable processes in a variety of different, higher-level applications. Initially ASP was supposed to resolve this problem by promoting software as-a-service (Currie and Seltsikas, 2002). Yet, the delivery and integration of ASP offerings proved difficult, especially for complex systems (unlike collaboration tools, such as e-mail).

Whilst *pure-play*[2] ASPs, having adopted the Web Service architecture, were better positioned than enterprise ASPs migrating traditional enterprise architecture technology to the Internet, the lack of flexibility and integration severely affected the progress towards the software as-a-service model (Hagel, 2003).

The promise of Web services

Evolving from object and component computing and implementing the OO middleware concept, Web services promise to bridge the gap between previous computing technologies. Web services are emerging to provide a systematic and extensible framework for application-to-application interaction; built on top of existing Web protocols and based on open XML standards (Curbera *et al.*, 2002). The communication problem in objects and components computing can be addressed by the standard interface mechanisms of Web services, instead of individual EAI solutions. For instance, currently, the communication between the COBRA architecture and Microsoft COM architecture is almost impossible since it needs a very complicated interface mapping approach to bridge the CORBA object and COM object (Pope, 1998). Moreover, there is no common standard interface except expensive proprietary EAI solutions. It has already been mentioned that Web services are in effect Web sites for computers to use. However unlike the human centred applications that need to consider presentation as well as content, these Web services only exchange information. This is known as a simple object access protocol (SOAP) document. This document is transmitted across the Internet using the standard HTTP protocol, and it is constructed with XML.

The structure of service orientated architectures (SOA)

The Web services standard comprises of four key technologies:

(1) XML, the universal format for structured data
(2) Universal Description, Discovery, and Integration (UDDI, www.uddi.org), a service description level that describes the network and eventually publishes and distributes this into the UDI
(3) Web services Description Language (WSDL, www.w3.org)
(4) SOAP, the application message protocol

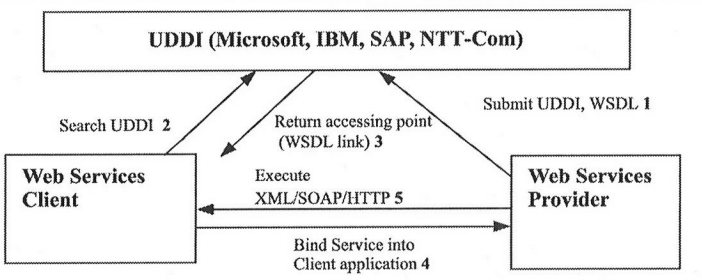
A typical SOA procedure as shown in Figure 1 consists of the following interactions (Vinoski, 2002):

(1) A Web Service advertises its WSDL definition into a UDDI registry; (Step1)
(2) The client looks up the service's definition in the registry and retrieves the WSDL link from UDDI; (Steps 2 and 3) and
(3) The client binds the service by the WSDL definition to sends messages or requests directly to the service via SOAP (Steps 4 and 5).

A comparison between Microsoft's .NET and Sun's J2EE platforms

The Microsoft .NET[3] platform and Web services are tightly integrated, which has given .NET an initial lead as a Web services platform. In contrast

Figure 1 Description and discovery processes in Web Service architecture



The Journal of Enterprise Information Management
Volume 17 · Number 5 · 2004 · 335–350

to the J2EE[4] framework which is a set of open standards, the Microsoft .NET framework is a product suite. Some of its offerings have been built on standards, others have extended these standards in a proprietary way. J2EE was designed to simplify complex problems with the development, deployment, and management of multi-tier enterprise solutions. J2EE is an industry standard, and is the result of a large industry initiative led by Sun Microsystems. Figure 2 shows some key differences in features between these two mainstream technologies (Chappell and Jewell, 2002; Lee and Geralds, 2002; Samtani and Sadhwani, 2002a, b).

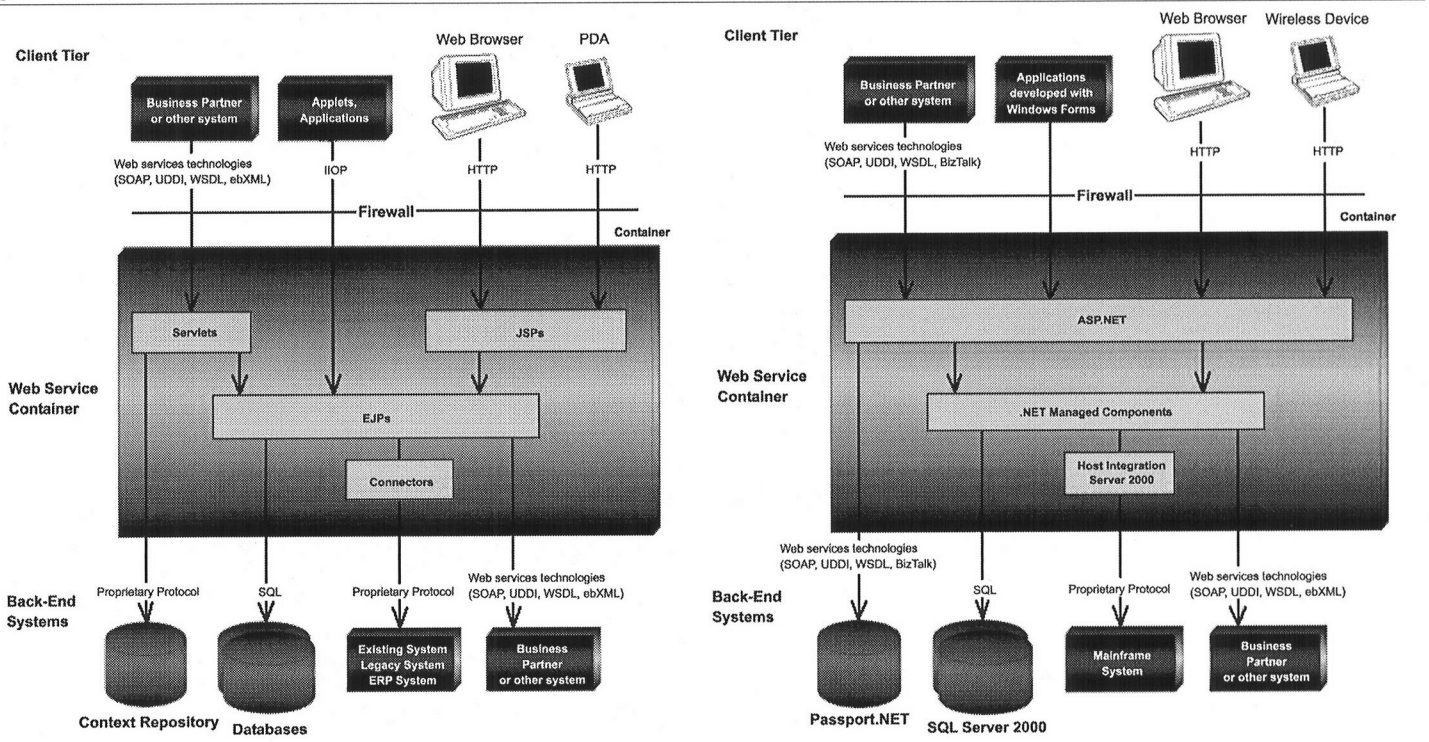
Fundamental design and support for Web services
.NET provides runtime support for SOAP, WSDL and UDDI as native .NET protocols. Tightly integrated it allows the building of XML-based Web services, and all Web services that communicate to each other via using ASP.NET. Support for Web services in Java is implementing through many APIs such as Java API for XML Messaging (JAXM), Java API for XML Processing (JAXP), Java API for XML Registries (JAXR), and Java API for XML-based RPC (JAX-RPC). The EJB+Servlet/JSP container will be the interpreter for J2EE's version Web services.

Business process management (BPM) and E-commerce
.NET provides business process management and e-commerce capabilities in its BizTalk server. These capabilities are not provided in a J2EE specification, but can be acquired from other Java technology vendors.

Support for existing systems
.NET offers legacy integration through the Host Integration Server 2000 or BizTalk server 2000. This can use the Window DNA architecture to communicate to the mainframe system. Supporting and integrating packaged applications and legacy systems (e.g. Siebel, Oracle, or SAP) will be easier with J2EE and J2EE Connector Architecture (JCA) rather than .NET.

Implementation
.NET claims to target an application integration between platforms using XML. All program code in .NET is compiled down to MS Intermediate Language (MSIL), regardless of whatever development language is chosen. This is not x86 machine instruction code, because if it were it would only run on processors that support this language. The MSIL is not dependent on any processor and needs to be further compiled into the native code that the processor understands. This is called the Common Language Runtime

Figure 2 The Web Services structure of J2EE and.NET. (**Source:** Vawter and Roman, theserverside.com)



The Journal of Enterprise Information Management
Volume 17 · Number 5 · 2004 · 335–350

(CLR). Unlike .NET which does not target a specific platform, J2EE has been created to focus on application portability and connectivity between platforms supporting Java. All Java Codes are compiled into bytecode, run by Java Runtime Environment (JRE). Thus, the implementation of Web services in J2EE will typically be done through Enterprise JavaBeans (EJBs).

Tools and server

Microsoft's cornerstone development tool, the Integrated Development Environment (IDE) for Web services is Visual Studio.NET (VS.NET). The Web services enabled servers from Microsoft including BizTalk 2002 and SQL Server 2000. In contrast to .NET, there are multiple companies that have built IDEs and application servers based on J2EE. Products such as Sun ONE Web services Platform, IBM WebSphere serials.

Maturity of platform

Although .NET inherits a lot of features from the Window DNA architecture, it is still relatively new and has to prove itself to be able to offer an enterprise-wide framework. J2EE has proven to be a robust, scalable and a mature platform over the last four years. The addition of support for Web services is just another feature for this platform.

Drivers of platform

Software companies are more likely to favour J2EE because it can run on a variety of different platforms. But users may prefer .NET because software can be developed in several programming languages, even though it only runs on Microsoft technology.

Both of these technologies have advantages and disadvantages. As a new Microsoft technology supported by its integrated development toolset Visual Studio.NET, it comprises the requirements for the development of the next generation of Web services. However, from a service providers' perspective, .NET is still relatively immature, particularly in regard to the integration with legacy systems. The adoption of .NET has also been restricted as most services' applications were developed by the J2EE standard, which is not supported by .NET. The technical challenge for Web services will therefore depend upon the issue of standards, and how legacy systems can be fully integrated with these new technology platforms and architectures.

Web services: the business challenge

The fallout from the first phase of ASPs, with numerous vendors going out of business, provided a salutary lesson for the market readiness of Web services. The biggest single factor, which led to the demise of start-up ASPs, was the lack of attention to how business value would be created for the customer (Currie, 2003). Many ASPs simply believed that potential customers would be convinced about the benefits of accessing their business software using a remote delivery model. But on closer analysis, it is difficult to see what benefits would be realised from a change in usage. For example, the notion that all customers would benefit from being able to access their software applications on a 24 × 7 basis was unfounded (ref). Many customers had little need for 99.999 per cent uptime, and were more concerned about data security and integrity than simply access (ref). Empirical findings from a recent study on customer satisfaction from ASPs indicated "a need for ASPs to facilitate integration with existing IT client organisations, ensure superior performance delivery, emphasize rigorous enforcement of SLAs, and ensure that their applications meets standards of software capability" (Suslarla *et al.*, n.d.).

These findings confirm the need for xSPs to provide significant business value to customers, manifested through EAI and possibly business process outsourcing (BPO) initiatives. Web services vendors, with their promises of integration, may therefore, succeed, where ASPs largely failed. But the hype surrounding Web services echoes that of ASPs a few years before.

According to Hagel (2003, p. 19), "What makes Web services technology so powerful is its distinctive ability to help managers operate more flexibly and collaborate more successfully with business partners".

Unlike ASP, which Hagel describes as a "false start", Web services promise to resolve the problems of lack of integration of applications and lack of flexibility. Some believe that Web services will create new, and transform existing, business and customer-facing software applications. Unlike traditional applications, which are stand-alone and self-contained, deployed in fixed locations like thick-client desktop machines or centralised application servers, Web services are not tied to specific technology platforms and operating systems. Applications based on delivery via Web services will benefit the customer through its simplicity of use within a single IT structure.

> Out of the ashes of the first wave of e-commerce experimentation, a new technology architecture is emerging. This technology architecture, known as Web services architecture, responds to major challenges that previous generations of technology have been unable to address. Despite its name, the focus of Web services architecture is not to connect people with Web sites. Instead, it focuses on connecting applications and data directly with each other, automating connections that might otherwise have required human intervention.

The Journal of Enterprise Information Management
Volume 17 · Number 5 · 2004 · 335–350

> The architecture is designed to ensure that applications and data can be accessed by authorised entities, regardless of location or underlying technology platforms (Hagel, 2003, p. 24).

The benefits of accessing software applications across multiple entities and heterogeneous technology platforms, at an affordable cost, are perceived as the key business benefit of Web services. The loose coupling of resources means that the "connections can be established without being tailored to the specific functionality embedded in the applications to be connected".

The business challenge for vendors, according to Gartner (13 Nov 2002, p. 2) is to provide:

(1) service orientated necessities, such as the ability to expose functionality as a service; and
(2) XML-based application programming interfaces (APIs), if not fully fledged SOAP APIs. Some of these features will come only with version upgrades. In rare cases, vendors will provide bonus functionality or new software through the use of Web services.

The xSP landscape is complex, involving a number of vendors, all of which contribute to the delivery of software-as-a-service. Whilst the core competencies of different xSPs are positioned in one of the main service provider categories, there may be overlap in the types of services provided by xSPs as shown in Table I.

A case study on Web services at Amazon.com

In order to examine the ubiquitously adopted potential of Web services worldwide, we discuss a case study using Amazon Web Service as a cooperated associate. Whilst Web services are still immature, lessons may be learned about the development and deployment of the client side Amazon Web service. In this section, we discuss the two steps of consuming a Web service, discovery and execution and illustrate how to create an Internet based customer client application to acquire the information from Amazon Web service provider.

According to recent research results (Baker, 2002) three clear steps are needed for Web services integration:

(1) internal integration;
(2) going outside the enterprise; and
(3) the collaborative agile business on the Internet.

Most of Web services today are remaining largely under the inside-the-firewall phenomenon. A big exception to this trend is Amazon Web services, which has initiated a public Web services program that lets Web site owners and developers to include Amazon features and information on their sites (Gralla, 2002). Amazon's Web Service efforts are a result of its popular associates program, in which website owners earn 15 per cent sale commission for each transaction they send back to Amazon. Initially, Amazon supported some static links containing the associate's ID which can be embedded in the associates' website. Although this program was quite successful, associates wanted more functionality from Amazon. For instance, instead of leaving the associates' website for searching and purchasing, they required more flexibility to complete those transactions staying in their own website.

Web services have emerged as an explicit way to solve the above issue. Either by using simple XML formats transacted over HTTP protocol, or embedding XML data into SOAP, associates can retrieve data from Amazon's Web Service provider that can then be formatted to display Amazon's product information on their site. Furthermore, it translates shopping cart transactions within the associates site. Associates can use XML "heavy" or XML "lite" documents when creating their Web services. The "heavy" documents give access to everything displayed on Amazon's product page, including customer reviews, images of the items to be purchased and more, while the "lite" document gives only a stripped-down number of features.

Table I xSP categories

Capacity service providers (CAPs)	Managed service providers (MSPs)	Application service providers (ASPs)	Internet business service providers (IBSPs)
Co-Location providers Telcos Wholesale service providers	Infrastructure platform providers Managed components providers	Managed applications providers (ISV or third party)	Develop and deliver Net-native application services
Resell Capacity ASP Integrators	Manage infrastructure	Outsource	Service
Aggregate and integrate services from multiple SP and ISV partners			

Source: Adapted from Summit Strategies, Inc

Discovery and integration of the online Amazon Web Service

The discovery of a Web Service will typically be done on a business-to-business level. In other words, two businesses will agree to exchange the data using Web services, and the provider of the service will inform the consumer about the way to find the service. The process to discover Web services will be dealt on the Universal Description, Discovery, and Integration service (UDDI, www.uddi.org) and its four member companies, Microsoft, IBM, SAP and NTT-Com, which provide the search engine services. Registered business members can publish their available Web services in WSDL format on UDDI for those potential customers.

In order to explore the appropriate service in our project, we searched Microsoft and IBM UDDI with the provider's keyword "Amazon". It would then respond with a WSDL document link (http://soap.amazon.com/schemas/AmazonWebServices.wsdl) containing the URL for invoking the XML Web Service, the Web methods, input arguments, and the output data type. As a built-in XML Web Service development environment, Microsoft VS.NET allows the developer to reference other Web services without any obvious complications. For instance, developers utilise Web service listener automatically mapping classes and functions of Web services providers instead of creating a proxy class manually as needed in other Web Service development platforms. Using "add Web reference" function in VS.NET, the platform itself will create all related classes and functions. Therefore, developers can browse the Web methods and parameters simply from "Class View" option, which speeds up the whole development process.

To retrieve the information from the Amazon Web Service provider, a Dataset object has been used to deliver the information. In ASP.NET, the process of creating and filling Dataset object easily simplifies complex transactions between database server and remotes client application. Binding the filled Dataset object into a DataGrid, (a special format to display information in HML page in the presentation layer), the client's application can illustrate all information by users' querying.

Lessons learned as Amazon Web services' client

Several important lessons have been learnt from this case. Together with quality criterions used to judge a web Service (Olsina *et al.*, 1999; Offutt, 2002), these can be summarised into three areas: maintainability, copyright and quality of service.

Maintainability

Although all elements to build up the Web services are not totally new technologies and these primary technologies have emerged as worldwide standards, Amazon Web Service cannot afford a robust service due to multi-supporting needed for .NET developers, J2EE developers and open source world PHP/Perl developers. Seamless interoperability across heterogeneous systems, platforms, applications and programming languages is not an easy task to be conquered. For instance, it may take several days for a support team to figure out the problems caused by .NET version Amazon Web Service.

Copyright and business model

There are not many appropriated business models and business laws for Web services. Amazon Web Service appears more as an extended service for its online shop. In terms of services level, it is not a pure service provider who credits its benefits upon its service transactions. Therefore, the business model of Web services is still immature. On the other hand, most existed Web services' clients are depending on the same Web services resource. However, the solution to avoid infringing each other's copyright could become a serious issue that needs to be resolved in the near future. A "Google-ish" Layout used for searching Web application which deploys Amazon Web Service has been banned based on similar issue. (Amazon lite, www.kokogiak.com/amazon/)

Quality of network service (QoS) in UDDI

Unlike normal Internet search engines, UDDI offers four search solution for users (Siddiqui, 2002). Hence, users can query UDDI Business Registry by business name, business location, business identifier, industry sector, service type, and other categories. However, the quality of results are not all of a similar standard (Zhang *et al.*, 2002). For example, a business search for "Online book store" in a popular search engine may respond with more than hundred online bookshop services, but may return nothing on Microsoft and IBM UDDI. On the other hand, the UDDI is only a basic directory. It is a simple database-like mechanism that lets participants insert descriptions of services they offer and query for services offered by other participants. There is not a brokering or facility service that is fully qualified to answer the customer's query intelligently (Huhns, 2002). For instance, in Amazon Web Service, the shopping cart facility can be easily integrated in the web application, however eventually the customer needs to finish the payment transaction on Amazon's Web site, but not on client's (associate) site. If a brokerage service could use knowledge about the

The Journal of Enterprise Information Management
Volume 17 · Number 5 · 2004 · 335–350

requirements and capabilities of registered service to decide an appropriate payment service, the whole process would be much appreciated by associates, and it also can help keeping the customers' loyalty on the associates' Web site, not Web Service providers'.

The research programme

The research study discussed in this paper originated in the UK and Silicon Valley, USA at the height of the dotcom era. Venture capital funding for start-up xSPs was relatively easy to come by in spite of poorly constructed business models with little detail on how they would generate revenues (ref). The overall aim of the research study was to develop a risk assessment framework for evaluating the deployment, hosting and integration of Web-enabled software applications by xSPs[5] and to develop a market segmentation database comprising the different product/service offerings of xSPs. The concept of Web services was under-developed at the beginning of the research study (in 1999). At the end of the 1990s, academic and practitioner literature was predicting that

(1) software-as-a-service would become a global phenomenon among SMBs providing new opportunities for the IT industry; and
(2) software-as-a-service would utilise a mix of capabilities and skills from xSPs (ASPs, ISVs, telcos, networking and co-locator firms).

To explore these themes, it was important to carry out a scoping study of the market readiness for xSPs to offer software-as-a-service. xSPs could offer both horizontal (business-facing) and vertical (industry-specific) software applications, so it was important to segment the xSP by market position and product/services portfolio. Key research questions were therefore, How do xSPs strategically position themselves to offer software-as-a-service? What is the product/service portfolio of xSPs? What is the customer value proposition used by xSPs? What KPIs do customers use to evaluate the benefits and risks of using xSPs?

To answer these questions, the research design comprised two stages. First, an exploratory qualitative study was developed to elicit rich data from respondents on both the supply-side and demand-side of software-as-a-service. The focus on vendors and customers was deemed essential, particularly as vendor-hype was distorting the true picture of how the xSP business models were being perceived by potential and existing customers. The study involved field research at vendor and customer sites, using a semi-structured questionnaire on how xSPs intended to create value for their customers by offering software-as-a-service. The results of this pilot study[6] led to the development of a questionnaire survey for evaluating the benefits and risks of using the software-as-service model delineated by five categories: delivery and enablement; management and operations; integration; client/vendor relationships; business transformation. Within each category, key performance measures (KPIs) were identified where respondents could rank their importance according to their own business requirements using a 1-4 scale. For this purpose, a Web-enabled database was developed to elicit responses from the questionnaire survey, though the initial data collection method was done manually by interviewing all respondents. This was to ensure reliability and validity of the research instrument and the responses received. It also gave respondents the opportunity to clarify the meaning of the range of questions and KPIs.

Second, a qualitative study was undertaken to develop a Web-enabled market segmentation database, comprising xSPs in three main categories: horizontal service providers (i.e. xSPs offering ERP, accounting, HR, travel and expenses software-as-a-service); vertical service providers (i.e. xSPs offering industry-specific – healthcare, travel, manufacturing, IT industry, software-as-a-service); and technology platform providers (telecos, networking, data centre, co-locators, etc.). The market segmentation database provided a search tool for customers and vendors comprising data on the strategic positioning; product/services portfolio; and value proposition of xSPs across the three main categories. For example, the market segmentation database would give a listing of all the xSPs providing accounting business software applications. ERP vendors offering software-as-a-service would also be listed. This data provides an overview of the xSP marketplace for potential and existing customers which may be used to

(1) identify a potential vendor;
(2) compare the product/service offerings of vendors; and
(3) match the offerings with the KPIs ranked in the xSP customer evaluation database discussed above.

Research results – developing Web services using the Microsoft.Net platform

This section discusses the technical development of the two Web-enabled databases[7]:

(1) The xSP customer evaluation tool; and
(2) The xSP market segmentation database

The Journal of Enterprise Information Management
Volume 17 · Number 5 · 2004 · 335–350

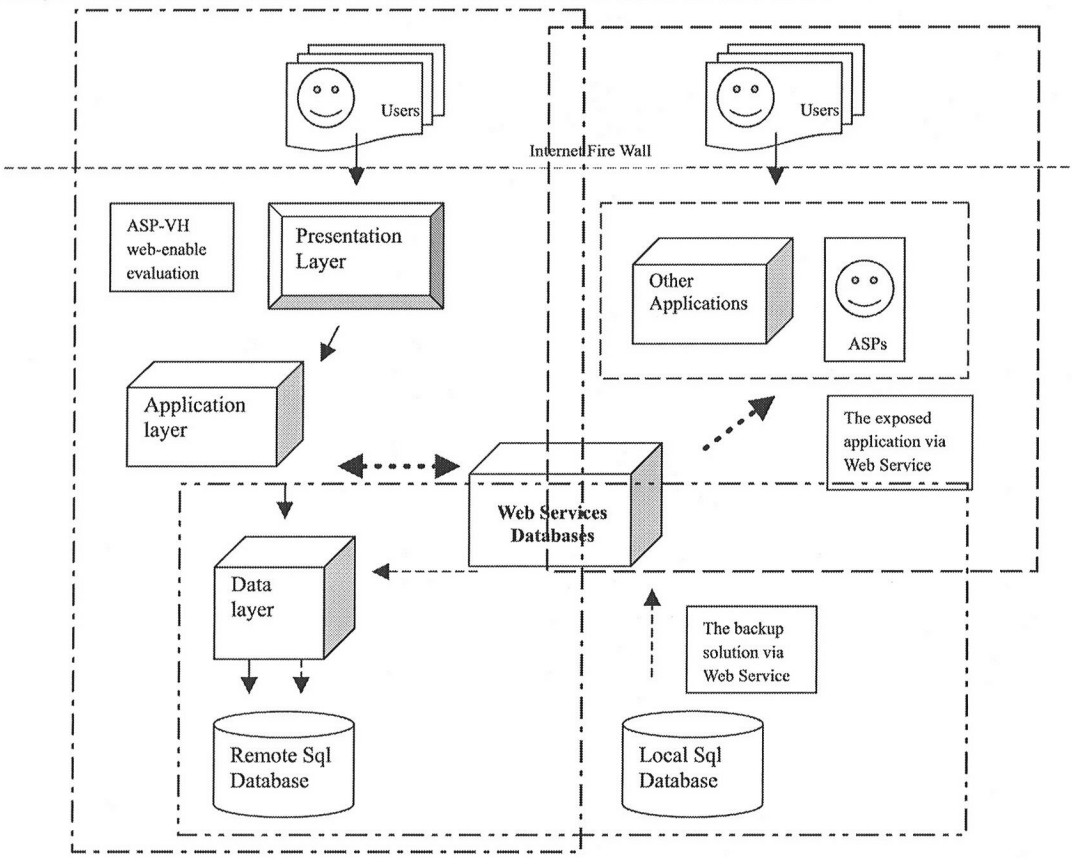
The use of the .NET platform and Web services

The databases were developed using the Microsoft .NET architecture which is shown in Figure 3. They represent a prototype for using Web services using SOAP/XML protocols within a Web services structure. As a Web hosting company hosts the Web-enabled databases, the approach to data backup is to retrieve the online data through a Web Service WSDL listener. The decision to use the .NET platform was twofold. First, the development of the two databases was not restricted by the need to integrate the applications with legacy systems. Second, the .NET platform is designed to encourage firms to develop and use Web services. There are many case studies available demonstrating using .NET for Web services (see http://Microsoft.com). Following a review of the data back-up options, a Web services back-up was implemented. All living data on the remote database server can be transferred to a local server and be filtered on daily basis. Both databases can be explored as online Web services. The use of the .NET architecture, using Windows 2000 as the operating system, SQL Server 2000 as the database to store relational data, ASP.NET as the server-side technology for creating dynamic Web pages, VS.NET and .NET framework SDK (shipped with VS.NET) as the .NET development tools, and VB .NET as the programming language for writing code in ASP.NET.

The ASP customer evaluation tool

One of the key objectives of the funded research studies was to provide customers (mainly the SMB sector) with a valuable tool to evaluate the benefits and risks of using an ASP, and also to gain an understanding of the product/service offerings of xSPs, and their strategic positioning. SMBs can access the Web-enabled tool and evaluate the importance they attach to KPIs across five categories: delivery and enablement; integration; management and operations; client/vendor partnerships; and business transformation. It is outside the scope of the present study to discuss all the categories. This paper will therefore, give the example of one category: delivery and enablement. For example, if an SMB was currently using/or planning to use, an ASP, they can use the ASP customer evaluation tool to evaluate the KPIs

Figure 3 ASP-VH Web Services architecture



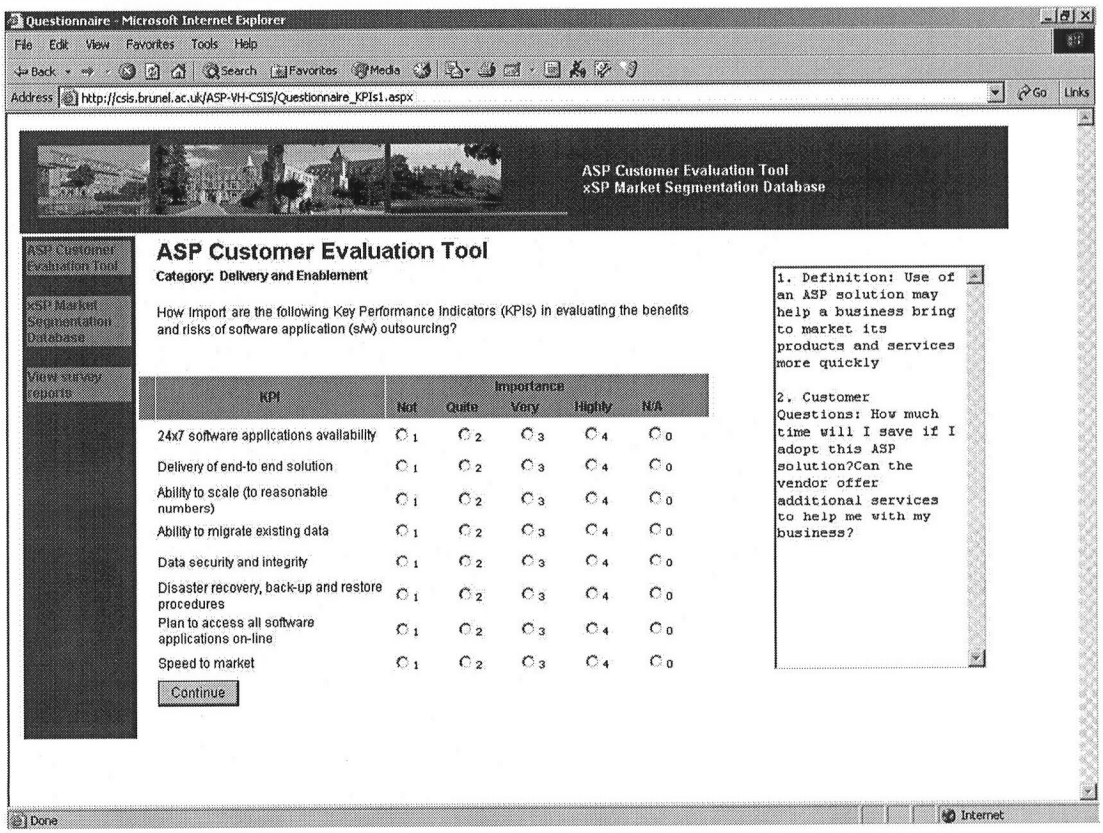
within the category of delivery and enablement. As Figure 4 shows, there are seven KPIs listed from "24 × 7 software application availability" to "speed to market". An SMB may decide that 24 × 7 software availability is a highly important KPI in using an ASP. Conversely, "speed to market" may only be considered quite important. Although these KPIs are generic, the SMB can apply them to specific software applications. An SMB using an ASP for enterprise resource planning (ERP), may therefore evaluate the KPIs differently from deploying a simple software applications, such as a travel and expenses package. The benefits and risks to the SMB from using ASPs for different software applications will therefore vary according to the importance or lack of importance given to each KPI across the five categories. The ASP customer evaluation tool further allows SMBs in specific vertical sectors (e.g. health) to compare their results with the aggregated results within this sector.

The technical development of the ASP customer evaluation tool is described below.

Presentation layer (front-end interaction)
Most Web script languages to program applications focus on the capability for rapidly creating simple functionality rather than modular construction of large programs (Knight and Dai, 2002). But unlike Microsoft's traditional Active Server Pages and other Open source script languages (PHP, Perl) combining server code and the client content intermixed in front page scripts, the presentation layer in ASP.NET has been separated from the backend business and application functions layer to clarify the development process. Utilising Microsoft VS.NET with OO components, the design task of ASP.NET has been significantly streamlined. Given that integrated common JavaScript functions and HTML functions are built into Web Forms, a new technology in ASP.NET, we can rapidly develop cross-platform, cross-browser programmable Web applications. However, this is still done with OO components running behind the presentation layer (Anderson *et al.*, 2001). Through our experiment, all objects in the presentation layer within business or application functions, should be chosen with built-in Web forms whether it is a simple validation process or a complex application process.

The code behind layer (application layer, mid-end integration)
As outlined in the presentational layer, properties and events for the server functions are processed in

Figure 4 ASP customer evaluation tool Internet interface (presentation layer)



The Journal of Enterprise Information Management
Volume 17 · Number 5 · 2004 · 335–350

the new concept, Code Behind, which is completely removed into a separate file from the presentation layer. The data from the presentation layer is encapsulated into the DataSet format in the Code Behind layer. Furthermore, with the session facility, which is an object to maintain data for a limited duration of time, the DataSet object can be accessed through the whole life cycle. Owing to consistency and security requirements of ADO .NET (API to database in .NET, see Data Layer section), we cannot submit the unfinished data in the DataSet into the database. Hence, we need to use the session variable to record the temporary data in each Web interface, and reload them into the DataSet in the final stage. A major technical issue has been addressed in this process, since we create a typed DataSet that has an existed data structure in it. In ASP.NET, the session state has been improved with new features. Storing data in the session variable can either be in the same memory that ASP.NET uses or in a separate memory from ASP.NET, or even in SQL server persistent storage. Thus, as users get routed to different servers, each server is able to access the user's Session data. The VB .NET as a new OO programming language has inherited a significant feature of the OO concept, the Component Architecture. Technically, a .NET component is a self-contained unit of functionality, with external interfaces that are independent of its internal architecture (Anderson *et al.*, 2001). To deal with some reusable functions, we programmed several re-useable functional components. Each represents the application function to the data table in the database respectively, so that they can be accessed from different parts of the application. For instance, the application function to fill in the *COMPANY* (a data table to fill the company details) data into one DataSet is re-useable for both adding-new-records and for modifications when reviewing/monitoring data.

The Web Service layer (back-end infrastructure support)

The Web Service module is composed of a SOAP/HTTP listener, one or more classes that implement the Web services business logic, and a metafile for the Web Service written with WSDL (UDDI is implemented on Internet level for the metafile). The SOAP/HTTP listener can read SOAP requests sent through HTTP or some other standard Internet protocol. The listener, an ASPX page in this project, passes the XML-based SOAP request and makes the desired call on the object(s) implementing the Web Service's business logic. The metafile contains a complete description of the Web Service encoded with WSDL. These languages could be considered as the SOAP equivalent of a COM interface definition language (IDL), and the metafile itself as the SOAP equivalent of a type library. VS.NET uses the information in the WSDL file to generate a proxy, a helper class that simplifies calling the Web Service.

The first level Web Service – preparing an appropriate feedback for users

The business logic behind the first level Web Service is a risk assessment module. The expected parameter (business sector ID) is passed by a Web method (inherited from the Web Service layer) from the Code Behind Layer to the Web Service Layer. After retrieving relevant sample data from the proper SQL Server 2000 data table, and executing several built-in statistic functions with the parameter, the Web Service itself will feedback a DataSet object to the Code Behind Level. These include all general average data in our sample, and segmentation data by business sector tied into the DataSet. Although the loose couple concept is illustrated as the best functionality Web services can offer to future Internet applications, in our project, we found that sometimes, .NET cannot handle the changes in the Data Layer correctly into the Web services Layer. For example, the modification in a store procedure in the SQL server could incorrectly change the sequence of WSDL in the Web Service layer and a changing of an element's name in the Web Service could crash the client application without updating web reference in the client's application.

The second level Web Service – conducting an online backup solution and a Web-enabled application

Two functions have been realised by invoking Web services as we mentioned above. We designed a remote back-up solution to transfer data for the hosting database server to our back-up dataset through Web Service mechanism (Figure 3). Also the questionnaire functionality has been explored to authorised xSPs (see dashed box without point in Figure 3). Those two features are currently in the experimental stage.

The data layer (database, back-end infrastructure support)

In the data layer, we designed a SQL database consisting of five tables using SQL server 2000. This is comprised of one primary key table, COMPANIES, which contains the data related to the individual contestants company details; four foreign key tables ASP_SERVICES, which is about usage of the ASP model by respondents; OUTSOURCED_APPS, which is about outsourcing by respondents; COMPANIES_PROF, which is saving confidential financial data of respondents; and

The Journal of Enterprise Information Management
Volume 17 · Number 5 · 2004 · 335–350

USERS, which is about to authenticate user role. The first three foreign tables are referenced by the company unique identification ID (Figure 5).

The connection object in this case is explicitly designed DataSet for data access independent of any data store. Owing to database operations being manipulated frequently through the application, the disconnected storage DataSet object is implemented to give more convenient access to multiple tables, rows and columns. The DataAdapter object is selected as the bridge between DataSet and database source. The API for data layer in this project is ADO .NET. It accesses the data information in a disconnected manner to ensure that data conversion is accurate and complete.

The xSP market segmentation database

As the research programme is both customer and demand-side facing, an xSP market segmentation database was developed to complement the ASP customer evaluation tool described above. As the technical specification of the ASP customer evaluation tool has already been described, this section will give an overview of the main purpose of the xSP market segmentation database. As Figure 6 shows, the database contains fields which capture data on "Vertical sector"; "products and services"; and "technology partners". In the example below, the vertical sector of banking, insurance, and finance is featured under the "vertical sector" category. If a user enters a "search", they will be given all the xSPs that provide software-as-a-service in this sub-category. Similarly, under the category "products and services", the user can select from a range of software applications under various sub-categories (applications, general or unspecified; accounting, HR, ERP, etc.). The category of technology partners allows users to search for firms in various sub-categories (telecoms, data centre providers, etc.).

The database provides a useful search tool for potential customers and vendors for comparing the different offerings of xSPs. This may be done to list all the ASPs, for example, which offer healthcare software-as-a-service solutions, or to provide a basis for comparing different e-business models within the same vertical sector. For example, SMBs thinking of adopting a software-as-a-service remote delivery model for payroll can search for all the xSPs which develop this type of application. They can do this by searching under the "vertical" category of banking, insurance or finance or the "products/services" category of "accountancy". The vertical search will elicit all those vendors who serve customers in banking,

Figure 5 ASP customer evaluation database structure

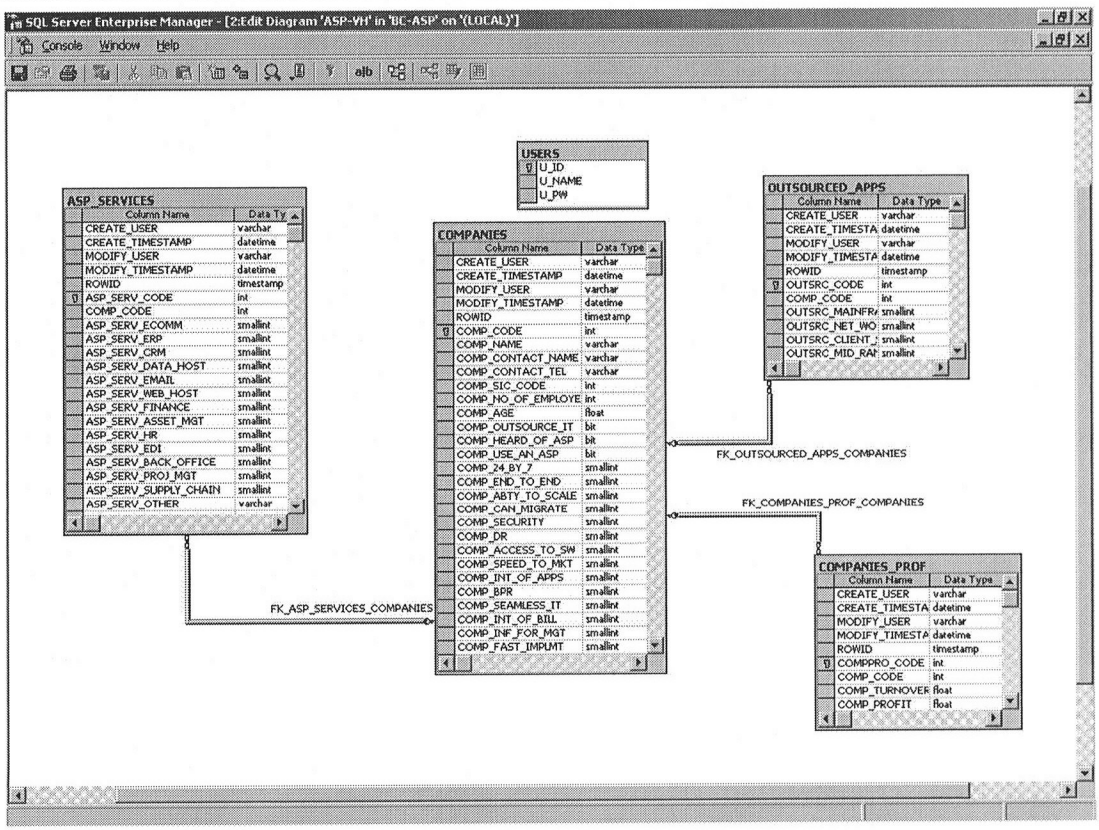
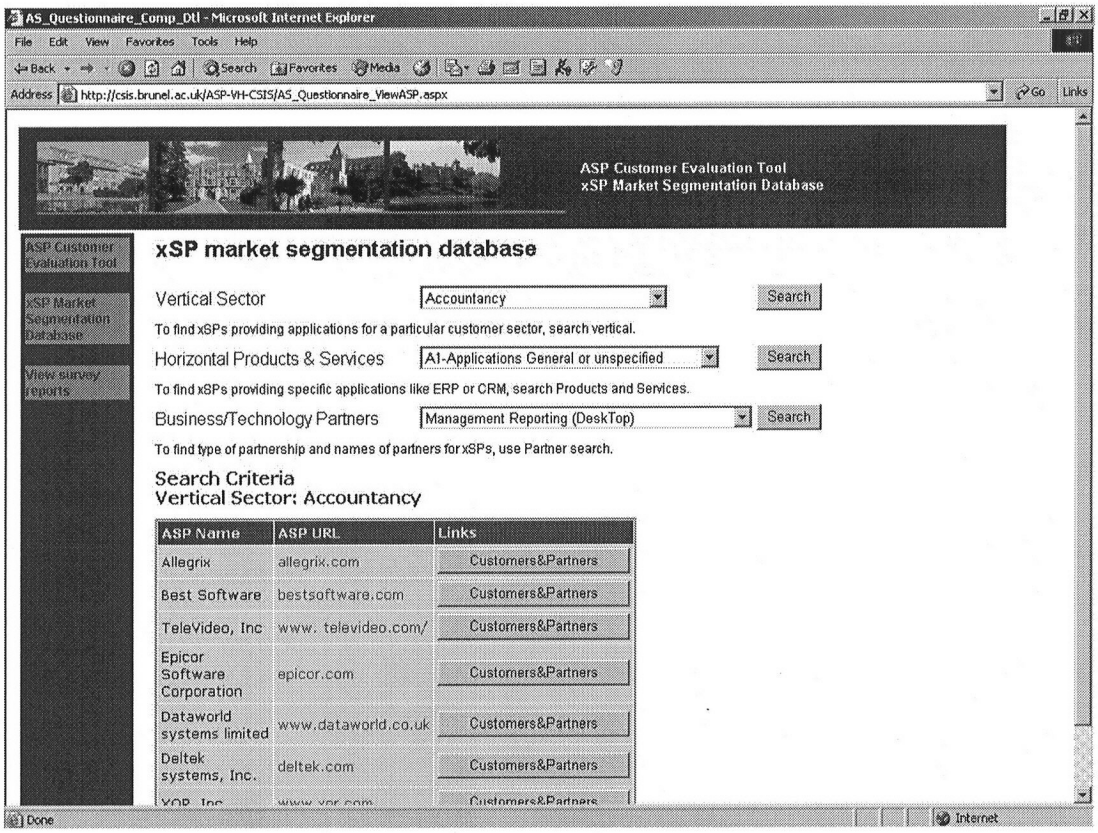


Figure 6 xSP market segmentation database



insurance and finance, but will not give too many details about the types of products/services offered. The product/services search will elicit all those vendors who develop accountancy software-as-a-service, so this search will be more appropriate for SMBs looking for specific applications and the xSPs who provide them.

The xSP database is designed in the first instance as a prototype to house data and information on xSPs in three categories (vendors; products/services; and technology partners). SMBs can use the database to undertake an initial search for vendors, and then build on this through their own research since all the vendors give the URL. The database is a useful tool for research purposes, especially since it contains "live" data on the current xSP marketplace. Changes in the scale and scope of offerings from xSPs can also be tracked over time.

Summary

In this section, we have discussed the preliminary results of the research programme, which has both a technical and business dimension. At the technical level, the .NET platform is used to develop two Web-enabled databases. At the business level, these databases provide both a customer and vendor facing perspective on the development of the software-as-a-service business model. The ASP customer evaluation tool allows SMBs to evaluate KPIs across five categories to assess the benefits and risks of adopting a remote software application delivery model. The xSP market segmentation database captures data and information on xSPs to give an overview of the industry structure. This allows SMBs (and vendors) to compare and contrast the product/service offerings of vendors.

Conclusion

This paper gave an overview of Web services and discussed how this technical development may resolve some of the problems of the first phase of the ASP market by integrating software applications across heterogeneous technology platforms and business environments. The findings suggest that business requirements will drive integration standards and that industry XML standards and capabilities are key to the success of Web services. To optimize the business potential of Web services requires the extension of features of a service environment such as UDDI or XML to be

The Journal of Enterprise Information Management
Volume 17 · Number 5 · 2004 · 335–350

introduced to industry-specific standards. As with ASPs, the ability to offer Web services is dependent upon partnerships with technology and service firms and industry demand for integration services will generate opportunities for Web services. Customers will need to be able to evaluate the benefits and risks of the software-as-a-service model and understand how the xSP industry is structured, and the products/services available. The research programme aims to assist customers and vendors in evaluating some of the benefits and risks inherent in the software-as-a-service model. Through identifying and understanding potential benefits and risks, users of ASPs can evaluate the products and services of xSPs, with the aim of including risk factors in service level agreements (SLAs). Clearly, the technical and business challenges for xSPs are significant, but the lessons learned from the fallout of the dot.coms and the ASP market suggest that vendors need to develop a technology-pull strategy, where customers demand the software-as-a-service model, rather than a technology-push approach, where products/services are developed without any recognition of how value will be created for the customer.

Notes

1 The generic term for service providers is xSPs. This incorporate service providers such as application service providers (ASPs) managed service providers (MSPs), Web Service providers, independent software vendors (ISVs) and many others.
2 A pure-play ASP is usually a start-up that develops applications to run on the Internet. Unlike enterprise ASPs, which are common built using client server technology, the pure-play ASP does not support "legacy technology".
3 Microsoft .NET is a set of Microsoft software technologies for connecting your world of information, people, systems, and devices. It enables an unprecedented level of software integration through the use of XML Web services: small, discrete, building-block applications that connect to each other – as well as to other, larger applications – via the Internet. (www.microsoft.com/net/basics/whatis.asp)
4 The JavaTM 2 Platform, Enterprise Edition (J2EE) defines the standard for developing multitier enterprise applications. (http://java.sun.com/j2ee/overview.html)
5 The initial research study focused upon the ASP business model as the unit of analysis as ASPs held a central position in delivering software-as-a-service, unlike telecoms and networking firms, which were largely IT infrastructure providers.
6 From the initial pilot study, research funding has been obtained from the Engineering and Physical Sciences Research Council (EPSRC) for a study on "Assessing the benefits and risks of business critical information systems using application services providers (BC-ASP); and from the Economic and Social Research Council (ESRC) for "A study on vertical and horizontal ASP business models" (ASP-VH). Total project cost for these studies is approximately £645,000, including £200,000 to industrial collaborators.
7 These databases can be accessed at: www.brunel.ac.uk/depts/cs/research/CSIS/asp-vh.htm.
8 Available at: http://Microsoft.com

References

Anderson, R. and Francis, B. *et al.* (2001), *Professional ASP.NET*, Wrox Press Ltd, Birmingham, AL.
Aoyama, M. and Weerawarana, S. *et al.* (2002), "Web services engineering: promises and challenges", paper presented at the 24th International Conference on Software Engineering, Orlando, FL.
Baker, S. (2002), "The three steps to Web Service integration", IONA.
Chappell, D.A. and Jewell, T. (2002), *Java Web services*, O'Reilly and Associates, Sebastopol, CA.
Curbera, F. and Duftler, M. *et al.* (2002), "Unraveling the Web services Web, an introduction to SOAP, WSDL, and UDDI", *IEEE INTERNET*, pp. 86-93.
Currie, W. (2003), "A knowledge-based risk assessment system for evaluating Web-enabled application outsourcing projects'", *International Journal of Project Management*, Vol. 21 No. 3.
Currie, W. and Seltsikas, P. (2002), "Evaluation the application service provider(ASP) business model: the challenge of integration", paper presented at the 35th Hawaii International Conference System Sciences, Hawaii, HI.
Ekanayaka, Y., Currie, W. and Seltsikas, P. (2002), "Delivering enterprise resource planning systems through application service providers", *Journal of Logistics and Information Management*, Vol. 15 No. 3, pp. 192-203.
Gralla, P. (2002), *Lessons from the Front: How Amazon is using Web services*, available at: searchwebservices.com.
Hagel, J. (2003), *Out of the Box, Strategies for Achieving Profits Today and Growth Tomorrow through Web services*, Harvard Business School Press, Boston, MA.
Huhns, M.N. (2002), "Agents as Web services", *IEEE Internet*, pp. 93-5.
Knight, A. and Dai, N. (2002), "Objects and the Web", *IEEE Software*, pp. 51-8.
Lee, C. and Geralds, J. (2002), "Web services wins customers at last", *Computing*, London, pp. 14.
Lim, B.B.L. (1996), "Component computing and objects: is there any common ground?", paper presented at the Tenth Annual Midwest Computer Conference, Loyola University Chicago, IL.
Offutt, J. (2002), "Quality attributes of Web software applications", *IEEE SOFTWARE*, pp. 25-32.
Olsina, L. and Godoy, D. *et al.* (1999), "Specifying quality characteristics and attributes for Web sites", paper presented at the 1st ICSE workshop on Web Engineering, ACM, Los Angeles, CA.
Orfali, R. and Harkey, D. *et al.* (1999), *Client/Server Survival Guide*, Wiley, New York, NY.
Pope, A. (1998), *The CORBA Reference Guide*, Addison Wesley Longman Inc, Reading, MA.
Samtani, G. and Sadhwani, D. (2002a), "Enterprise application integration (EAI) and Web Sevices", in Fletcher, P. and Waterhouse, M. (Eds), *Web services Business Strategies and Architectures*, Expert Press Ltd, Birmingham, AL, pp. 39-54.
Samtani, G. and Sadhwani, D. (2002b), "Web services and application frameworks (.NET and J2EE)", *Web services*

Business Strategies and Architectures, Expert Press Ltd, Birmingham.

Siddiqui, B. (2002), "UDDI-based electronic marketplaces", in Fletcher, P. and Waterhouse, M. (Eds), *Web services Business Strategies and Architectures*, Expert Press Ltd, Birmingham, AL.

Stal, M. (2002), "Web services: beyond component-based computing", *Communications of the ACM*, Vol. 45 No. 10, pp. 71-6.

Suslarla, A., Barua, A. and Whinston, A.B. (n.d.), "Understanding the service component of application service provision: an empirical analysis of satisfaction with ASP services", *Management Information Systems Quarterly*, Vol. 27 No. 1, pp. 91-118.

Vinoski, S. (2002), "Web services interaction models part1: current practice", *IEEE Internet Computing*, pp. 89-91.

Zhang, L-J. and Li, H. *et al.* (2002), "XML-based advanced UDDI search mechanism for B2B integration", paper presented at the Fourth IEEE International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS'02), Newport Beach, CA.

Further reading

Adams, H. and Gisolfi, D. (2002), *Best Practices for Web services: Back to the Basics, Part 1*, IBM developer Works.

Currie, W. (2004), "Value creation from application service provider e-business models: the experience of four firms", Enterprise Information Management (forthcoming).

Currie, W. and Seltsikas, P. (2001), "Delivering business critical information systems though application service providers: the need for a market segmentation strategy", *International Journal of Innovation Management*, Vol. 5 No. 3, pp. 323-49.

Glass, G. (2000), "The Web services (r)evolution, applying Web Servies to applications", *IMB: developerWorks, Web services Library.*