Adoption Challenges in Migrating to Web Services

Scott Tilley
Department of Computer Science
University of California, Riverside
stilley@cs.ucr.edu

John Gerdes
Graduate School of Management
University of California, Riverside
john.gerdes@ucr.edu

Terrance Hamilton
Department of Computer Science
University of California, Riverside
thamilton@cs.ucr.edu

Shihong Huang
Department of Computer Science
University of California, Riverside
shihong@cs.ucr.edu

Hausi Müller
Department of Computer Science
University of Victoria
hausi@cs.uvic.ca

Ken Wong
Department of Computing Science
University of Alberta
kenw@cs.ualberta.ca

Abstract

This paper outlines some of the challenges inherent in migrating to Web services. Technology adoption is a continual challenge for both tool developers and enterprise users. Web services are a prime example of a red-hot technology that is fraught with adoption issues. Part of the problem is separating marketing hype from business reality. Web services are network-accessible interfaces to application functionality. They are built using Internet technologies such as XML and standard protocols such as SOAP. The adoption issues related to Web services are complex and multifaceted. For example, determining whether this technology is a fundamental advance, rather than something old under a new name, requires technical depth, business acumen, and considerable historical knowledge of past claims. A model problem from the health care industry is used to illustrate some of the adoption issues that arise in migrating to Web services.

Keywords: adoption, migration, SOAP, technology transfer, Web services, XML

1. Introduction

Web services are network-accessible interfaces to application functionality. There are many articles, journals, and Web sites that promote the use of Web services in almost all possible situations. The result is that there is so much hype surrounding Web services that it is quite difficult to distinguish marketing claims from technical innovation and business reality. The promise of Web services, the truly novel types of applications that it enables, and the many vendor-specific commercial offerings are jumbled together in such a manner as to make even the most determined technology tracker confused.

Nevertheless, Web services really do offer a novel approach to engineering and deploying software solutions such as cooperative information systems. For example, there is a growing trend towards the integration of networked heterogeneous applications across the enterprise, a phenomenon that can be greatly facilitated through the judicious use of Web services.

The promise of Web services must, however, be tempered by the very real challenges faced by both developers and users of Web service-enabled applications. For example, the skill set that must be mastered by a software engineer to properly develop an application that relies on Web services is considerable. Similarly, the deployment costs of these applications can be significant, both in terms of new software systems and new hardware and networking infrastructure.

This paper does not attempt to present Web services as a software panacea. Instead, a balanced view of the promise and pitfalls of migrating to Web services is presented. While there is little doubt that Web services do represent a technical advancement, they are a means to an end, not an end in themselves. Moreover, the advancements are more evolutionary than revolutionary – current trade press articles notwithstanding.

The next section provides an overview of the fundamental technologies underlying Web services. Section 3 discusses three particularly acute adoption issues related to Web services: separating marketing hype from technical reality; the skills needed for modern software engineers to make proper use of Web services in developing new applications; and the challenges in migrating legacy systems to a Web services platform. To

Service	Implementation
Discovery	UDDI (Universal Description, Discovery and Integration)
Description	WSDL (Web Services Description Language)
XML Messaging	XML-RPC (Extensible Markup Language-Remote Procedure Call), SOAP (Simple Object Access Protocol)
Transport	HTTP (Hypertext Transfer Protocol), BEEP (Blocks Extensible Exchange Protocol (BEEP))
Network	TCP (Transmission Control Protocol)
	IP (Internet Protocol)

Table 1: Web Service Protocol Stack (adapted from [7])

illustrate some of these issues, Section 4 presents a model migration scenario of a deployed Web-based system. Section 5 summarizes the adoption issues discussed in this paper, and offers advice on navigating the Web services waters in the near future.

2. Web Services

Browser-based applications are now commonplace. However, as useful as the Web browser is for users surfing the Internet, it is difficult for an automated service to make use of the same facilities that a person can use. For example, online travel sites offer a wide selection of services for booking airline tickets, reserving cars, and selecting hotels. But it is not easy for a programmer to write the code needed to process the HTML that the travel site produces in response to user's free-text queries. This is true even when tools for gathering data from HTML pages (e.g., WebL [10]) are used.

It would be simpler if the site exposed its services (possibly for a fee) to automated agents, so that application-to-application communication takes place over the Web, not just user-to-application [16]. This is the promise of Web services: network-accessible interfaces to application functionality [11].

2.1 What Are Web Services?

A Web service is a special type of a peer-to-peer (P2P) application, and the canonical example of modern net-centric computing (NCC). Web services can be viewed as a step along the road from the current Web infrastructure to the so-called "semantic Web," a richer and more intelligent World Wide Web being developed with guidance from the original Web's inventor, Tim Berners-Lee, under the auspices of the W3C [22].

One definition of a Web service is "… a network-accessible interface to application functionality, built using standard Internet technologies" [25]. Another definition of a Web service is "… any service that is available over the Internet, uses a standardized XML messaging system, and is not tied to any one operating system" [7]. In both definitions, the notions of network accessibility, platform independence, and standards-based service are paramount. All Web services share the desirable characteristics of self-description and the ability to discover other services with which to cooperate.

For a human user, self-description of a Web service might mean documentation that makes it easier to integrate the Web service into a larger application. For another Web service acting as a client, this might mean a machine-processable interface written in a common XML grammar. The XML grammar can be used to identify all publicly-available methods, their signatures, and their return types.

To facilitate cooperation among different Web services, a discovery mechanism is required. A Web service, even if it is novel and unique, would be useless if it cannot be found on the network. This implies the needs for a place where the description of the Web service can be published. Similarly, there needs to be a way to search and query this repository for a particular Web service.

One way of thinking about Web services is from a role perspective. In this view, a Web service is composed of a service provider, a service requestor, and a service registry. In true P2P form, a Web service can serve one, two, or all three roles in an integrated system simultaneously. Such a service is called a "servent" (i.e., both server and client) in P2P terminology [15].

2.2 The Web Services Protocol Stack

Another way of thinking about Web services is from a protocol perspective. Like most network applications, Web services can be viewed as a layered stack of protocols. For Web services, the stack consists of the service transport layer, the XML messaging layer, the service description layer, and the service discovery layer. This perspective of Web services is illustrated in Table 1.

A fifth composite layer can be considered to implicitly exist at the bottom of the stack: the underlying network layer. This is the same TCP/IP-based mechanism used by the rest of the Internet to provide basic communication, addressing, and routing capabilities.

The transport layer is responsible for sending messages between applications. The main transport layer for Web services is currently the hypertext transfer protocol (HTTP), the same protocol used by Web browsers. This is one of the most attractive aspects of Web services, because it means the enhanced functionality of a Web service can immediately leverage the considerable network infrastructure already in place. By relying on HTTP as the transport protocol, Web services can avoid some of the deployment problems that plagued earlier middleware technologies, such as CORBA and DCOM [6].

The XML messaging layer is responsible for encoding messages in a common XML format so that the message can be understood by all communicating services, irrespective of hardware platform or implementation language. The XML-RPC specification provides a simple message format that encodes remote method invocation as XML tags [28]. Its use is quite limited when compared to the more common XML messaging mechanism, the Simple Object Access Protocol (SOAP) [23]. The power of SOAP is that it relies on regular XML to provide a mechanism to invoke a Web service and to retrieve the results in a consistent manner. The entire SOAP message is wrapped in an XML envelope and sent using HTTP.

The service description layer is responsible for describing the public interface to a specific Web service. This is currently implemented with the Web Service Description Language (WSDL), although other techniques (such as DISCO from Microsoft [20]) have been proposed by industrial participants in Web services. As with other aspects of Web services, WSDL is an XML grammar. The public interface to a Web service can include all methods provided by the service (and their invocation requirements), binding information to specific transport protocols, and address information for locating the service. Using WSDL, a client can locate a Web service and invoke any of its available functions. With WSDL-aware tools, this process can be automated, enabling an application to easily integrate new services with little or no manual coding, thereby reducing the need for low-level programming, as would be required for control integration through a scripting language.

The service discovery layer is responsible for centralizing Web services into a common registry. This registry is logically a single entity, but it might be realized as a decentralized service. Service discovery is currently achieved by the Universal Description, Discovery, and Integration (UDDI) mechanism. Data in UDDI is stored in XML format. UDDI acts much like the Domain Naming System (DNS) on the Web, or like a telephone book for users. Through UDDI, users and other Web services can locate Web services and, once found, retrieve the Web service description in WSDL format, and then invoke the service via SOAP.

2.3 The Promise of Web Services

Web services hold the promise of considerable gains for many organizations [18]. For example, one of the long-standing goals of the software engineering and information systems communities has been to develop techniques to effectively integrate disparate applications by leveraging computer technology. Rather than acting as independent programs, integrated systems can provide better business value by sharing data, communicating results, and improving overall functionality. The challenge has always been how to realize this goal. By using a standards-based and vendor-neutral approach that relies on Web services, this goal is closer to reality.

However, it should be said that previous generations of middleware technology came with similar promises. One of the most important differences between Web services and its predecessors is momentum. The earlier solutions relied on proprietary software. They also lacked a widely-available network infrastructure. In contrast, Web services are being developed by the community at large, and they have the largest network ever constructed – the Internet – with which they can make themselves available.

Nevertheless, Web services technology is still somewhat immature, with the underlying technologies—XML, SOAP, WSDL, and UDDI—being relatively new. Clearly, more research is needed before migration via Web services will be commonplace [26]. Perhaps one of the most important places to begin such an investigation is the topic of adoption.

3. Web Services Adoption Issues

If one believes the many articles currently appearing in the popular technology press, it would seem that Web services are the greatest innovation ever. Not only can Web services be used in numerous situations to realize elegant solutions to pressing business needs, but they can be adopted by both developers and customers with little fanfare. As usual, the practical reality of the situation is somewhat different from this utopian techno-centric view of the world.

3.1 Marketing Hype versus Business Value

From the business perspective, the allure of Web services stems primarily from the promise of savings, either through centralization of these services within the organization, or through outsourcing agreements. Within the organization's Intranet, the Web services model allows the information technology (IT) department to regain control over the IT resources of the firm. Using a centralized model improves version and access control while integrating "islands of data." It can also improve application quality and consistency by placing the development and maintenance responsibility with IT professionals, rather than leaving it in the hands of IT enthusiasts. The IT department necessarily has a broader perspective than individual departments, and thus can better address the long-term needs of the organization.

The other alternative is to outsource part or all of the IT functions. Based on a resource-based view of the firm, it is those resources and capabilities controlled by the firm that are unique, rare, imperfectly imitable, and non-substitutable that creates a competitive advantage [3]. It is these resources that constitute the firm's core competencies. By definition, those activities not falling within this set can be done more efficiently by someone else. Outsourcing those functions of the firm in which it has no competitive advantage allows the firm to focus on what it does well. The same benefits given above for centralizing Web services within the firm may be realized through outsourcing.

Companies that are in the business of providing Web services will likely be able to do a better job than those not in that business. They have the experience and know the potential pit falls. There is also a benefit to be derived from the "network externalities" of a larger client base [2][9][17]. Experience from other applications and clients will likely improve the application's overall operation. Direct costs may also be reduced due to the increased efficiency of the outsourcing agent. Development costs can be spread over a large, global customer base, thereby reducing the costs even further.

There are drawbacks with this approach. Whenever responsibility shifts to a third party, there is a loss of control, both in timing and functionality. The priorities of the Web service provider may not be aligned with that of the firm, and therefore implementation of desired features may be delayed, and certain functionality may never be implemented.

From a business perspective there are many dimensions that must be examined when considering the move to a Web services model. These include issues of cost (both short term and long term), timing, flexibility, control, maintenance, support staff, and return on investment to name but a few. A 2001 IDC study of 54 application service provider (ASP) customers, found that firms are reaping significant returns from Web services—on average, over 400% return over 5 years [19]. This optimistic assessment must be tempered with the recent GAO report on desktop outsourcing within 6 federal agencies. Although positive results such as better IT management and desk-help support are claimed, "GAO could not determine whether any of the agencies were achieving expected cost benefits because they did not perform sufficient up-front analyses of their baseline and projected costs and benefits or routinely monitor all actual seat management costs and benefits" [14]. The report's conclusions are equally relevant to industry as they are to the agencies involved:

- Agency commitment is crucial, especially by top management
- Thorough up-front preparation and planning activities must be completed
- Solicitation and contract award activities should be carefully managed
- Program and contract management activities, including monitoring contractor performance, are key
- Partnerships between agencies and the [outsourcing] contractors, in which they have common goals, are critical
- Effective and continual communication within the agency, with the outsourcing contractor, and among contractors working on related activities.

3.2 The Renaissance Revisited

The paper "On the Emergence of the Renaissance Software Engineer" [27] stated the following:

In the past, one could prosper with basic knowledge of algorithms and data structures, programming languages, and rudimentary project management. In today's net-centric world, the skills required by a software engineer engaged in Web site evolution activities have increased dramatically. New areas of expertise, such as distributed component technology, computer security, and Internet standards like XML, are becoming core knowledge areas. To be successful, one must become a renaissance software engineer, with knowledge that is both broad and deep. This implies keeping abreast of both research in academia, developments in industry, and watching governmental policy.

With the advent of Web services, the situation has only been exacerbated. There are so many new technologies for today's developers to learn, and so little time to learn them, that it seems nearly impossible to keep

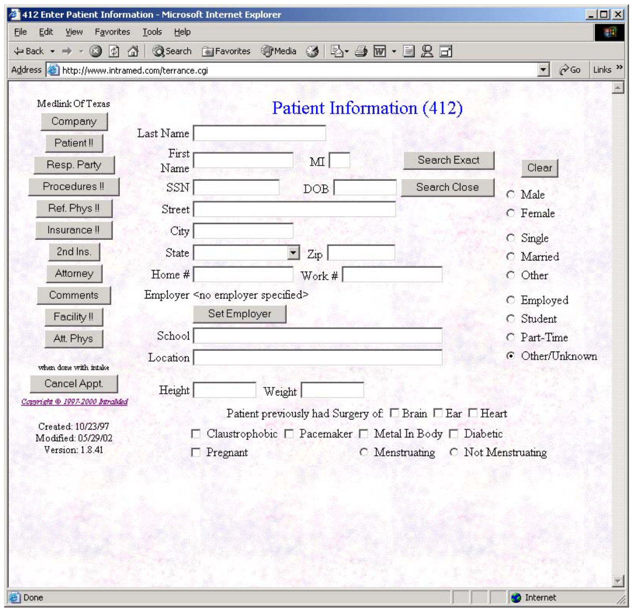

Figure 1: Patient Tracking Wizard

up. Even the meta-knowledge that one needs to determine the lack of knowledge in a new area is difficult to cultivate.

If there is one thing about Web services that is unequivocally true, it is acronym rich: .NET, API, ASP, DISCO, EJB, HTTP, IIS, J2EE, NCC, ONE, SOAP, SQL, TCP, UDDI, VBA, WSDL, and XML are just some of the technology monikers that are directly related to Web services. While lengthy, this list grows considerably when supporting technologies, such as ActiveX, C#, and Java are added. Not only do developers need to know the meanings of these acronyms, they need to know how to make proper use of the technologies in order to engineer software solutions that are reliant on Web services. It is true that several of these technologies are quite similar to previous technologies, but even being able to notice this requires extensive historical knowledge on the part of the developer.

3.3 Legacy Web Systems

Much has been written about the importance of business processes that are embedded in today's deployed legacy systems. Reverse engineering [8] can be used to extract these business rules as a prelude to migrating legacy systems to modern platforms. However, the difficulties of using such technology should not be underestimated [21]. In fact, there is considerable resistance to modernizing legacy systems, due to the immaturity of the technology used, and the uncertain business value of such a costly migration effort.

The last few years have seen considerable attention devoted to providing Web interfaces to traditional legacy systems, which are typically characterized as monolithic, single-tier, mainframe-based applications.

The investment required to carry out such migrations were considerable, both in terms of educating the developers in newer technologies (e.g., network computing) and convincing customers of the benefits (e.g., extensible applications). Now with Web services, the same businesses are being asked to redevelop their applications. The difference is that the first migration was from monolithic to Web-based, and this migration is from Web-based to Web service.

4. A Model Migration Scenario

To illustrate some of the adoption issues faced when migrating to Web services, a model problem from the health care industry is used. The sample application is a claims tracking system called the Patient Tracking Wizard (PTW). The PTW is a browser-based scheduling, billing, and fax system for a medical network, independent practice association (IPA), third party administration (TPA), or other similar organization. An image of the PTW application is shown in Figure 1.

The PTW system is chosen as a model problem because it has characteristics that are representative of legacy Web applications. It has a front-end that is browser-based and more functional than elegant. It has a back-end that relies on older Web technologies such as CGI scripts to connect the application to the database. As described in 4.2, the PTW is in need of modernization to remain commercially viable.

The entire PTW program can be run locally or remotely over the Internet. It monitors the status of a patient, including initial scheduling, appointment day, doctor's reports, paying vendors, billing the payers, and billing the patient. The PTW assigns incoming faxes to the various cases and automatically notifies insurance companies, providers, and referring physicians when cases are scheduled and when reports come in. Management reports of several types are easily generated. Electronic data interchange (EDI) interfaces with payers and finance companies are available.

4.1 Current System

The current legacy system is implemented using the C programming language and a proprietary 4GL language called HCC. As a low cost solution, it is primarily deployed on a Linux server with an Empress database [13] backend. The HCC code is an abstraction of HTML 1.0 that describes a Web page in terms of rows, thus supporting simple forms and tables. HCC also supports the Set as its only collection data type. HCC supports the application presentation; the C code is primarily used to interface with the database.

A build consists of several iterations compiling 242 source code files. The HCC (20KLOC in 63 source files) is compiled to C code and appended to one large target C code file which is approximately 67KLOC. The C and header files (40KLOC in 99 C and 78 header files, excluding the files generated from HCC) are compiled and linked to form a relatively small and fast CGI binary. State in the application is maintained on the page with an invisible form element that is encrypted.

The Web interface is the primary user interface to the application; however, it also supports a suite of client/server tools that can be downloaded and installed on a PC. These tools (wizards) were originally constructed using Visual Basic 6.0 (75KLOC) and support pre-canned reporting, fax identification, Health Care Financing Administration forms (HCFA) generation, itemized bill generation, and notification. The wizards are outdated and much of the code suffers from poor documentation, suggesting a prime candidate for reengineering.

4.2 A Web Services Migration Candidate

The PTW was designed and constructed as a low cost solution to track medical claims, workers compensation claims, and scheduling of resources. It also supports other business models such as IPA and TPA business scenarios. It is in use today as the primary billing tool of two very successful private corporations using these business models. Nevertheless, the PTW has never quite delivered its full potential as originally envisioned. The advent of Web services represents an opportunity to evolve the PTW application to make it even more flexible, powerful, and valuable to its users. In other words, move PTW closer to its original design goals.

One of the limitations of the current PTW system is that the scheduling feature requires a call center attendant to do all of the resource scheduling by phone. The wizards were originally designed to operate in-house only, requiring a company to purchase and install the system locally, which further requires an IT staff or a maintenance contract. The PTW is Health Insurance Portability and Accountability Act (HIPAA) compliant, but the distribution of the wizards in support of client/server interaction via the Web breaks this compliance. The fax wizard must have a mapped network drive to operate successfully. This could be achieved with a virtual private network (VPN), but becomes a problem if the server is installed offsite at a secure provider location. Secure providers typically do not allow connections less than direct T1 access, which results in higher operating costs. The PTW is also weak in the area of locale sensitivity and accessibility features.

Due in part to these limitations, the PTW seems like a prime candidate for migration to Web services. It has a

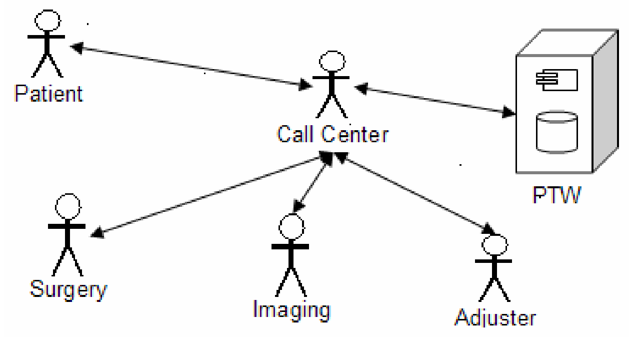

Figure 2: Current Call Center

number of potential stand-alone modules and features that could be provided as services in unique configurations. Good candidates for separation of services are resource scheduling, provider and patient search, patient claims tracking and follow up, notification and faxing, and a billing package to include dynamic form generation and EDI. The wizards could be eliminated or incorporated into the Web application instead of client software. Detailed reporting could be done with a commercial-off-the-shelf (COTS) product such as Crystal Reports [12]. The infrastructure for this exists and can be reused. Report and billing forms outside of a Crystal Reports type solution can be dynamically generated as Acrobat (PDF) files. This technology already exists as a Web service from Adobe and can be purchased for integration into the system or used dynamically via the Internet [1].

Another technology that exists today is enterprise resource scheduling software. Resource scheduling is one of the primary functions of the PTW in all of its supported business models. Typically there is a need to schedule resources such as nursing staff, beds, doctors, rooms and equipment, as well as patient appointments and product delivery. PTW's current method of scheduling resources (i.e., call center) could be enhanced with a Web interface that allows registered users to schedule appointments and resources via the Internet. This can be provided as a Web service to the PTW and its users.

The database which supports the resource schedule need not be integrated directly with the PTW interface. Exposing scheduling and notification services to the current PTW application will eliminate some of the call center issues and replace the notification wizard with a server-side email and e-fax notification system. In turn, the PTW can expose search capabilities and key business entities that support its collection of business models. Furthermore, by integrating a COTS server-based resource scheduling system, the PTW could also benefit with a much needed facelift. The COTS resource schedulers are customizable and provide many look and feel options. By taking advantage of the maturity of

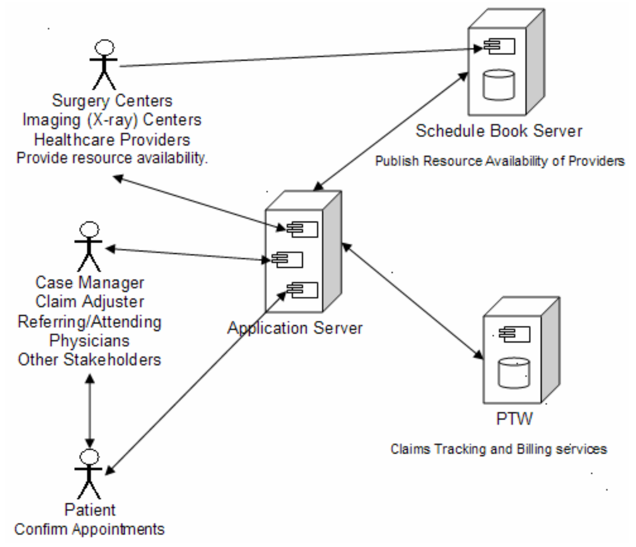

Figure 3: Proposed Call Center

HTML, XML, CSS, and other modernized technologies, the PTW will evolve into a more usable application.

4.3 Proposed Deployment Architecture

To illustrate some of the architectural changes that would be required by migrating to Web services, consider the current call center application. A simplified use-case diagram of the call center as it is currently designed is shown in Figure 2. All services are contingent upon having well-trained call center staff. The call center personnel are responsible for confirming all reservations and using the wizards to complete the remainder of services. Each arrow represents the flow of data. The call center representative must exchange information with all parties including the PTW.

The proposed Web services-based architecture provides a suite of applications that will help providers and payers manage patient claims and resources. Each node exposes specific Web services to be integrated in various application configurations. A simplified use-case diagram of the proposed call center's deployment architecture is shown in Figure 3.

A summary of the changes in supporting technologies between the current version of the PTW and the proposed Web services version is shown in Table 2. There are many advantages promised by the Web services solution, but there are also many disadvantages. The primary disadvantage is the risk involved in rebuilding the software system – a risk shared by many large-scale reengineering efforts [5]. Cutting-edge technology is considered risky because advances are frequent, and business has learned a hard lesson regarding new technology that does not create profit.

The task of rebuilding the system should not be underestimated. The legacy system is in very fast C code which is well understood, something that cannot be said with certainty for the Web services solution. Web services introduce a new paradigm in architecting software systems and a multitude of potentially new programming languages. A successful Web services deployment is dependent upon a firm grasp of XML and other languages such as the .NET family of languages (VB.NET, C#) or the Java Enterprise Edition specifications.

The prospective solution as presented requires purchasing COTS products to provide key functionality. There are many hidden risks associated with enterprise integration of COTS products that might not support the desired functionality in an exposed API. For example, the schedule server under consideration does not support Web services functionality, and must be generated from the data model or purchased separately from the vendor. The schedule server is highly configurable but may not accommodate all the required functionality. For the migration to Web services to be successful, all of these risks must be assessed and mitigated.

5. Summary

There are many misconceptions regarding the applicability and efficacy of Web services. Their proponents would have you believe that Web services are absolutely the most important new software development ever. Naysayers denigrate Web services as old middleware in new clothes – and transparent ones at that. As usual, the truth lies somewhere between these two extremes.

There is an important role to play for Web services in the development and deployment of modern software systems. The advantages of an open, standards-based, and vendor-neutral platform with which one can perform system integration across the enterprise should not be minimized. Web services hold the promise to a truly interoperable and heterogeneous application environment.

At the same time, the challenges in adopting Web services should not be overlooked either. From the developer point of view, there is a bewildering array of new technologies, buzzwords, and engineering paradigms encompassing such disparate areas as distributed components, databases, and networks. From the customer or service-provider point of view, there are numerous infrastructural issues that must be addressed before Web services become as ubiquitous as other operating systems services.

The largest barriers to Web services adoption may stem from the "grassroots" perceptions of developers and end users, as well as "top-down" business directions from administration and management. Issues of Web services' claimed technological superiority need to be balanced by

Old Client/Server Version	New Web Services Version
CGI Web application – single node • Application (HCC & C code) • Tightly coupled w/database using C	*N*-Tier Web Application (EAI) • Application(JSP, Servlet, EJB) • Scheduling Services (ASP/C#.NET) • Database accessibility through Web services
Call Center Scheduler: Schedule appointments by calling facility and patient.	Enterprise Scheduler on separate node, facility publishes schedule to Web service.
Fax identification and assignment with VB Wizard – Requires mapped network drive.	Build into Web application. No mapped drives.
Notification Wizard (VB)	Auto notification through email, included with Schedule Server and eFax.
Itemized Billing Wizard (VB)	Bill presentation via PDF or EDI via modem from server.
HCFA forms Wizard (VB)	PDF Rendering of HCFA forms.
Report Wizard (VB)	PDF rendering of detailed reports for internal use with COTS.

Table 2: PTW Technology Changes

pre-existing business conditions. In general, several perceptual factors contribute to the rate of technology adoption, including relative advantage, compatibility, complexity, visibility of results, and critical mass [24].

Web services technologies need to provide an advantage over previous technologies and provide a business advantage over competitors. Web services technologies need to integrate well with technologies already used in existing systems—systems that are mission-critical business assets. Web services should be simple to develop and deploy, with avenues to try the technology incrementally at smaller scales, and with results and benefits that are clear (such as increased business opportunity or reduced maintenance costs). Also, without a critical mass of other Web services providers and requestors, exposing one's system via a Web service interface would lead to little business benefit.

In the sample scenario discussed in Section 4, a Web-based system is considered as a candidate for migration to Web services. Done properly, there would be measurable benefit to such a migration. For example, by separating the functional components into identifiable services, the potential for more widespread use of the system would be enhanced. However, performing such a separation for an existing system is a non-trivial task. Even assuming it was accomplished, the learning curve and infrastructural requirements imposed by the introduction of Web services would be considerable.

Perhaps the true payback of Web services is still to come. Although technically interesting, Web services are nothing more than a new (albeit rather advanced) form of Internet plumbing. From a business perspective, the benefits of Web services will only be realized when novel applications are built using Web services. Whether these services are used in aid to improve access to legacy systems, or in developing new ones, the real impact of Web services is still to be felt.

References

[1] Adobe, Inc. *Adobe Acrobat Family*. Online at http://www.adobe.com/acrofamily/main.html.

[2] Au, Y.; Kauffman, R. “Should We Wait? Network Externalities, Compatibility, and Electronic Billing Adoption.” *Journal of Management Information Systems*, Fall 2001, 18(2): 47-63.

[3] Barney, J. “Firm Resources and Sustained Competitive Advantage.” *Journal of Management*, 17, pp. 99 – 120. 1991.

[4] Barney, J.; Wright, M.; Ketchen, Jr. “The Resource-Based View of the Firm: 10 Years After 1991.” *Journal of Management*, 27, pp. 625-641. 2001.

[5] Bergey, J.; Smith, D.; and Tilley, S.; Weiderman, Nelson; and Woods, Steven. *Why Reengineering Projects Fail* (CMU/SEI-99-TR-010). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, April 1999.

[6] Cappell, D. *Understanding .NET: A Tutorial and Analysis*. Addison-Wesley, 2002.

[7] Cerami, E. *Web Services Essentials*. O'Reilly & Associates, 2002.

[8] Chikofsky, E.; and Cross, J. "Reverse Engineering and Design Recovery: A Taxonomy." *IEEE Software* 7(1):13-17, January 1990.

[9] Choi, J.; and Thum, M. "Market Structure and the Timing of Technology Adoption with Network Externalities." *European Economic Review*, 42, 2 (February 1998), 225-244.

[10] Compaq Corp. *Automating the Web: Compaq's Web Language (WebL)*. Online at www.research.compaq.com/SRC/WebL.

[11] Coyle, F. *XML, Web Services, and the Data Revolution*. Addison-Wesley, 2002.

[12] Crystal Decisions, Inc. *Crystal Reports*. Online at www.crystaldecisions.com/products/crystalreports.

[13] Empress Software, Inc. *Empress Relational Database Management System*. Online at www.empress.com.

[14] General Accounting Office (GAO). "Desktop Outsourcing: Positive Results Reported, But Analyses Could Be Strengthened." GAO-02-329 March 29, 2002. Online at www.gao.gov.

[15] Gnutella. Online at gnutella.wego.com.

[16] Handler, J. "Agents and the Semantic Web." *IEEE Intelligent Systems*, March – April 2001, pp. 37.

[17] Katz, M.; and Shapiro, C. "Network Externalities, Competition, and Compatibility." *The American Economic Review*, 75, 3 (June 1985), 424-440.

[18] Kirda, E.; Jazayeri, M.; Kerer, C.; and Schranz, M. "Experiences in Engineering Flexible Web Services." *IEEE Multimedia*, January – March, 2001, pp. 58-65.

[19] Mears, J. "ASP Customers Hail Return on Investment" *Network World Fusion*, March 25, 2002,. Online at www.nwfusion.com/news/2002/0325specialfocus.html.

[20] Microsoft Corp. "DISCO Web Services Discovery Tool." Online at msdn.microsoft.com/library/default.asp?url=/library/en-us/cptools/html/cpgrfwebservicesdiscoverytooldiscoexe.asp.

[21] Müller, H. CASCON 2001 Workshop on Adoption-Centric Tool Development. November 7, 2001; Toronto, Canada.

[22] Port, O. "The Next Web." *Business Week*, March 4, 2002.

[23] Rodes, K. "XML-RPC versus SOAP." Online at weblog.masukomi.org/writings/xml-rpc_vs_soap.htm.

[24] Rogers, E. *The Diffusion of Innovations*. Free Press, New York, New York, 1983.

[25] Snell, J.; Tidwell, D.; and Kulchenko, P. *Programming Web Services with SOAP*. O'Reilly & Associates, 2002.

[26] The 4th International Workshop on Web Site Evolution (WSE 2002). Online at star.itc.it/wse2002.

[27] Tilley, S.; and Huang, S. "On the Emergence of the Renaissance Software Engineer." *Proceedings of the 1st International Workshop on Web Site Evolution* (WSE'99). Atlanta, GA: October 5, 1999.

[28] XML-RPC. Online at www.xmlrpc.com.