JOURNAL OF SOFTWARE MAINTENANCE AND EVOLUTION: RESEARCH AND PRACTICE
J. Softw. Maint. Evol.: Res. Pract. 2004; **16**:31–50 (DOI: 10.1002/smr.283)

Research

On the business value and technical challenges of adopting Web services

S. Tilley[1,*,†], J. Gerdes[2], T. Hamilton[3], S. Huang[3], H. Müller[4], D. Smith[5] and K. Wong[6]

[1]*Department of Computer Sciences, Florida Institute of Technology, Melbourne, FL 32901, U.S.A.*
[2]*Graduate School of Management, University of California, Riverside, CA 92521, U.S.A.*
[3]*Department of Computer Science, University of California, Riverside, CA 92521, U.S.A.*
[4]*Department of Computer Science, University of Victoria, BC, Canada V8W 3P6*
[5]*Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213, U.S.A.*
[6]*Department of Computing Science, University of Alberta, Edmonton, AB, Canada T6G 2E8*

SUMMARY

This paper provides a balanced perspective of the business value and technical challenges of adopting Web services. Technology adoption is a continual challenge for both tool developers and enterprise users. Web services are a prime example of an emerging 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 developments. A sample problem from the health care industry is used to illustrate some of the adoption issues. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: adoption; business value; migration; SOAP; technology transfer; Web services; XML

1. INTRODUCTION

Web services are network-accessible interfaces to application functionality. If one believes the many articles in the popular technology press, it would seem that Web services are the greatest information

*Correspondence to: Scott Tilley, Department of Computer Sciences, Florida Institute of Technology, 150 West University Boulevard, Melbourne, FL 32901, U.S.A.
†E-mail: stilley@cs.fit.edu

Received 1 December 2002
Revised 11 April 2003
Accepted 30 June 2003

technology innovation in the past decade. The unstated assumption is that Web services can be used in numerous situations to realize elegant solutions to pressing business needs, and that both developers and customers can easily use Web services with little effort. As is often the case with emerging technology, the practical reality of the situation is somewhat different from this utopian techno-centric view of the world. There is potential business value in using Web services to modernize existing software applications, but there are numerous technical challenges that must first be addressed before the benefits can be realized.

1.1. What are Web services?

One definition of a Web service is '... a network-accessible interface to application functionality, built using standard Internet technologies' [1]. 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' [2]. In both definitions, the notions of network accessibility, platform independence, and standards-based service are paramount. In this context, a Web service functions like a special type of a peer-to-peer (P2P) application, and is the canonical example of modern network-centric computing (NCC).

Web services can also 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 World Wide Web Consortium (W3C) [3]. The W3C recently published three important documents related to Web services: *Web Services Architecture, Web Services Glossary* and *Web Services Architecture Requirements* [4]. The definition of Web services from the W3C is '... a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols'.

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 description might mean a machine-processable interface based on a standardized XML schema. The schema can be used to identify all publicly available methods, their signatures, and return types.

To facilitate cooperation among different Web services, a discovery mechanism is required. A Web service, even if it were novel and unique, would be useless if it cannot be found on the network. This implies the need 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 play one, two, or all three roles in an integrated system simultaneously. A service that plays the role of both a server and client is called a 'servent' in P2P terminology [5].

Another way of thinking about Web services is from a protocol perspective. In this case, a stack of Internet-related network protocols describes the hierarchical structure of Web services. This perspective is detailed in Appendix A.

1.2. Separating hype from reality

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 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 a manner that makes even the most determined technology tracker confused.

Nevertheless, Web services 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 requirement 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 adopting Web services is presented. From a business perspective, there is clearly business value in a standards-based and vendor-neutral solution. However, Web services are a means to an end, not an end by themselves.

From a technical point of view, there is little doubt that Web services do represent an identifiable advancement in the area of middleware. However, the advancements are more evolutionary than revolutionary—current trade press articles notwithstanding. Moreover, there are substantial challenges that must be addressed before whole-scale adoption of Web services can be considered to be a sound business decision.

1.3. Paper outline

The next section discusses the potential business value that Web services represent. Section 3 balances this discussion by outlining some of the technical challenges of adopting Web services. To illustrate both of these issues, Section 4 presents a sample problem of using Web services in the modernization of an already-deployed Web-based health information system. Finally, Section 5 summarizes the main points of the paper, and comments on the future of Web services.

2. THE BUSINESS VALUE OF WEB SERVICES

Web services hold the promise of considerable business value for many organizations [6]. However, the potential benefits must be balanced with a careful examination of the many issues related to the move to a Web services business 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. As reported in [7], a 2001 study of 54 application service provider (ASP) customers estimates that firms will reap significant cost savings from Web services: on average, over 400% return over five years.

This optimistic assessment must be tempered with the recent U.S. General Accounting Office (GAO) report on outsourcing within six federal agencies. Although positive results such as better IT management and help-desk 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 management costs and benefits' [8]. The report's conclusions are equally relevant to industry as they are to the agencies involved.

This section describes two particularly important areas where Web services can truly provide business value. (This assumes that issues such as those discussed above are properly addressed.) The areas are enterprise application integration and business-to-business e-commerce.

2.1. Enterprise application integration

With 70% of the lines of source code in corporations dedicated to performing data integration (moving data between systems), the problem of sharing data and having information flow smoothly across the enterprise is a major one [9]. The proliferation of net-centric applications, Web-based systems, and heterogeneous environments has only made the problem more acute. The situation is further exacerbated when the scale of the attempted integration grows to encompass the entire organization. This has led to a renewed interest in the problem, now commonly referred to as enterprise application integration [10].

Finkelstein describes enterprise (application) integration as '… the process of analyzing the structure and content of databases to identify redundant data that exists within and across enterprises' [11]. While true, we subscribe to the broader definition of enterprise application integration espoused by the Software Engineering Institute [12], which states 'enterprise integration has the goal of providing timely and accurate exchange of consistent information between business functions to support both strategic and tactical business goals in a seemingly seamless manner'. In both cases, the fundamental problem of integrating selected components from possibly disparate applications remains the same [13].

Wasserman described the delineation of the problem of enterprise integration into the three categories of data, control, and presentation integration in the 1980s in a seminal paper focused on tool integration in software development environments [14]. In the early 1990s the problem was extensively studied under the guise of CASE tool integration [15]. Several different approaches were advocated, such as federated systems [16], but all had deficiencies when it came to ease-of-integration for users of the resulting conglomeration. Most of the integration solutions required extensive programmer involvement and/or changes to the individual tools involved in the integration effort. For example, scripts written in languages such as Tcl/Tk can be used to provide control and presentation integration in software reverse engineering environments [17], but the knowledge and skills needed to perform such low-level coding is typically beyond the capabilities of the average end-user. Moreover, most users are more interested in just using the tool—not reprogramming it.

Many existing applications in daily use by governmental organizations and most major corporations were originally built in a stove-pipe manner to address very specific needs [18]. Over time, the scope of requirements has increased, along with an unfortunate tendency for the information systems to become brittle, difficult to manage, and hard to understand. This in turn leads to the inability of users to integrate critical new applications into the existing solution set, or to mix-and-match capabilities provided by the systems to solve new problems.

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 were still very vendor-specific. Although middleware such as CORBA conforms to a non-proprietary standard, the actual implementations didn't interoperate well, so one tended to choose a specific vendor, with the corresponding vendor-lock problems that entails. Earlier solutions 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.

The use of network technologies to assist with the problem of enterprise application integration has been proposed by several people (e.g. Finkelstein and Aiken [19]), but usually in the limited context of data integration. Web services offer the possibility of full integration along all three vectors of data, control, and presentation. Once legacy assets are mined and wrapped appropriately into reusable components, Web services can enable integrated access to these components. This sort of 'just in time' integration (JITI) is one of the long sought-after goals of information systems research, and it is already beginning to be used in the commercial arena today [20].

For example, Intel is working with an IT consultancy to provide Web services to the Albertson's grocery store chain [21]. The goal is to replace the current manual process of gathering distributors' pricing changes, and provide Albertson's with automated analysis of stock items in real-time and in an automated manner. At first glance it may seem odd that a hardware-oriented company like Intel is interested in a software-oriented development like Web services, but Intel hopes to embed much of the processing of Web service protocols in their chips of the future. In other words, special hardware may handle SOAP messages just like routers currently handle TCP/IP packets.

2.2. Business-to-business e-commerce

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 [22]) 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 [23]. This is the promise of Web services: network-accessible interfaces to application functionality [24]. In other words, Web services can facilitate the transition from business-to-consumer (B2C) e-commerce to business-to-business (B2B) e-commerce.

There is considerable business value in providing such interfaces, since the potential users of an online system can be magnified many times, augmenting human users with machine clients of the service. B2B e-commerce can be facilitated by identifying those areas of the business that are considered core competencies, and those areas that can be outsourced. This sort of business componentization shares some of the same characteristics as the process of mining legacy systems for reusable software assets [25]. Once the core components are identified, they can be enabled as B2B services to be provided to other Web services.

From the business perspective, the allure of Web services stems 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 imitated, and non-substitutable that create a competitive advantage [26]. 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.

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 pitfalls. There is also a benefit to be derived from the 'network externalities' of a larger client base [27–29]. Experience from other applications and clients will likely improve the operation of the company's core services. 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 potential 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 requestor, and therefore implementation of desired features may be delayed, and certain functionality may never be implemented.

3. THE TECHNICAL CHALLENGES OF WEB SERVICES

As described in Section 2, there is potential business value in the judicious use of Web services. However, before the business benefits can be realized there are numerous technical challenges that must be addressed. Two of the most difficult issues to resolve are related to skills of the people who will be responsible for actually implementing the Web services-based solution (software engineering skills), and the existing applications that must be modernized to make full use of Web services (legacy systems).

3.1. Software engineering skills

Many people say that the world is becoming so complex that no one can know everything. This is, of course, self-evident. But many people take this to be true even in their chosen discipline. For example, in computer science, if someone is a ‘theory’ person, the common assumption is they cannot be a ‘network’ person too. The very nature of a PhD degree is to focus on a very narrow problem space; it becomes difficult to widen one’s perspective afterwards.

Yet today’s software applications are moving in the opposite direction. It is true that no one can know everything about a particular area. But to manage the sheer complexity of system construction and evolution in a Web services-based environment, one now needs to know a lot about a lot; ignorance of other areas is no longer a tenable position. There are now so many different aspects to application software that there is a (re)emergence of the renaissance person, one who is comfortable operating in several disciplines, moving readily between computer science, software engineering, and information technology.

A software engineer from a previous era needed to know about computer science issues such as algorithms and data structures, operating systems, programming languages, and so on. They also needed to be aware of non-technical issues such as project management, effort estimation, and risk analysis. With the advent of Web services, the situation has only been exacerbated. Today’s software engineer engaged in Web services development needs to know all these things—and more. One might ask why a competent C++ programmer should also need to be a network security expert and a database expert as well. The answer is that these are just some of the new skills required of the new renaissance software engineer [30].

However, there are so many new technologies for today’s software engineers to learn, and so little time to learn them, that it seems nearly impossible to keep up. Even acquiring the meta-knowledge surrounding new and emerging areas can be 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.

Consider the case of Microsoft .NET [31][‡], marketed (in part) as a platform for building Web services. It is surprisingly difficult to define .NET from a technical perspective. There is a lot of marketing jargon behind .NET that makes it hard for a software engineer to understand the essential aspects of .NET and to place .NET in a continuum of Microsoft and non-Microsoft middleware initiatives. For example, what are the differences between the .NET Platform and the .NET Framework?

[‡]The .NET Framework is the programming model of the .NET platform for building, deploying, and running XML Web services and applications. In the .NET Framework, all components can be XML Web services, where an XML Web service is just another type of Microsoft Component Object Model (COM) component. The .NET Framework itself consists of two main parts: the Common Language Runtime (CLR) and a unified set of class libraries. These libraries include ASP.NET for Web applications and XML Web services, Windows Forms for smart client applications, and ADO.NET for loosely coupled data access. ASP.NET includes design-time objects, controls, and a run-time execution context for developing and running server-side applications. Visual Studio .NET is an integrated development environment that supports a visual programming model for the creation of .NET Web services.

Can an existing ASP page work without change in ASP.NET? Where does Visual Studio.NET fit into the picture?

It is neither possible nor desirable to keep abreast of absolutely every single software trend. Some trends are really just fads, quickly fading away. Other trends, however, are truly important; Web services are one of these trends. It has the potential to create a dramatic change in the nature of software engineering as currently practiced. For today's software engineer, new areas of expertise, such as distributed component technology, computer security, and Internet standards like XML that form the foundation for Web services, are becoming core knowledge areas. To be successful, one must have knowledge that is both broad and deep. This implies keeping abreast of both research in academia, developments in industry, and changes in governmental policy.

3.2. Legacy systems

Much has been written about the importance of business processes that are embedded in today's deployed legacy systems. Although not considered interesting to work on by many software engineers, legacy systems represent valuable business assets that cannot simply be discarded when new technology arrives. Instead, legacy systems often undergo extensive reengineering to extend their productivity in response to changing user needs.

A typical legacy system can be characterized as a monolithic, single-tier, mainframe-based application. The last few years have seen considerable attention devoted to providing Web interfaces to traditional legacy systems. In such a project, reverse engineering [32] can be used to extract business rules that can then be re-implemented on the newer platform. A legacy system may be decomposed into separable modules by undergoing a three-step reengineering process. First, reverse-engineering techniques may combine static and dynamic analysis with various modularization criteria (such as coupling and cohesion measures [33]) to identify potential components. Second, the modules are developed into virtual components that can be accessed on the newer platform. Finally, the components can be deployed across the network for use in a new business context. The result of the three steps of decomposition, development, and deployment is a modernized set of legacy components ready for integration with newly developed or acquired applications across the enterprise [34]. These revitalized components can then function as part of a harmonized information system.

The investment required to carry out such migrations can be 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). Moreover, even in this traditional scenario, the difficulties of using such technology should not be underestimated [35]. In fact, there is often 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. Techniques such as the Options Analysis for Reengineering (OAR) can be used to guide the decision-making process [36].

With the current focus on Web services, the same businesses are being asked to again 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. The challenges in extracting sufficient detail from this new type of legacy system are substantial [37]. In fact, to fully understand the functional nature, high-level design, and implementation details of a Web-based system using traditional reverse engineering technology is beyond the current state of the art. From a business perspective, such a lack of technical capability may be viewed as an unacceptable risk.

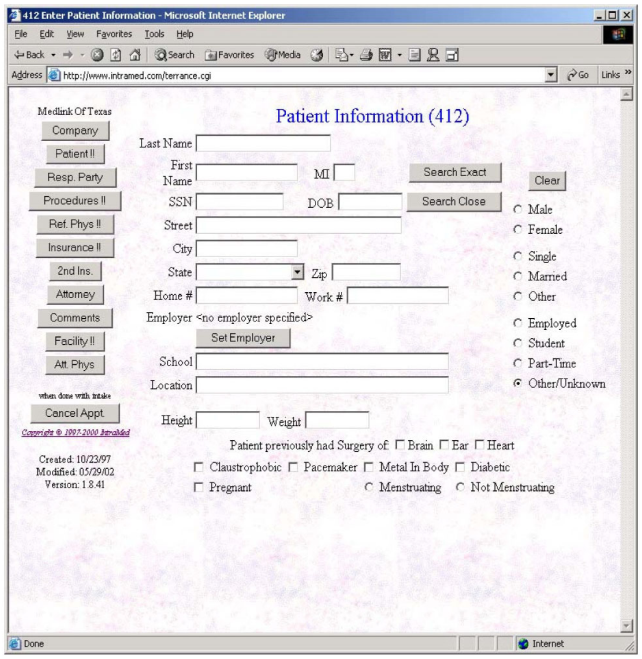

Figure 1. Patient Tracking Wizard.

4. A SAMPLE PROBLEM

To illustrate some of the potential business value and corresponding technical challenges faced when adopting Web services, a sample migration 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 Web-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 has characteristics that are representative of legacy Web applications. It has a Web front-end that is 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 Section 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 called HCC. As a low cost solution, it is primarily deployed on a Linux server with an Empress database [38] backend. The HCC code is an abstraction of HTML 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; C code is primarily used to interface with the database.

A system build consists of several iterations of compiling 242 source code files. The HCC source (20 KLOC in 63 source files) is compiled to C code and appended to one large target C code file that is approximately 67 KLOC. The C and header files (40 KLOC 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 a hidden 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 (75 KLOC) and support pre-scripted reporting, fax identification, Health Care Financing Administration (HCFA) forms 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 not realized its full business value 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 require at least T1 connections (1.544 Mbps), which results in higher operating costs. The PTW is also weak in the area of internationalization and accessibility features.

4.3. Business value in migrating the PTW to Web services

Due in part to the limitations described above, the PTW seems like a prime candidate for migration to Web services. It has a number of potential stand-alone modules and features that could be separated from the main application and provided as B2B services in several 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 as a separate client. Detailed reporting could be done with a commercial-off-the-shelf (COTS) product such as Crystal Reports [39]. 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 [40].

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. The 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 in the migrated PTW to 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 e-mail 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 modernization of its user interface. The COTS resource schedulers are customizable and provide many look and feel options. By taking advantage of the maturity of HTML, XML, CSS, and other modern technologies, the PTW will evolve into a more usable application.

4.4. 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 I. There is significant potential business value

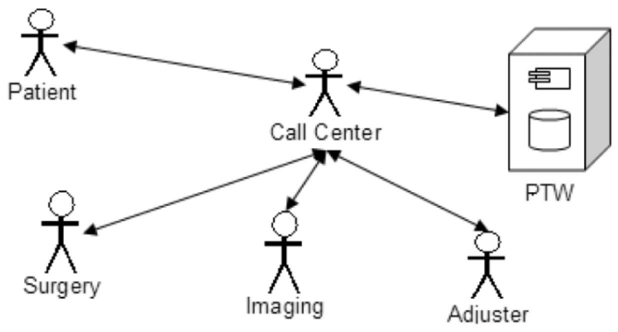

Figure 2. Current call center.

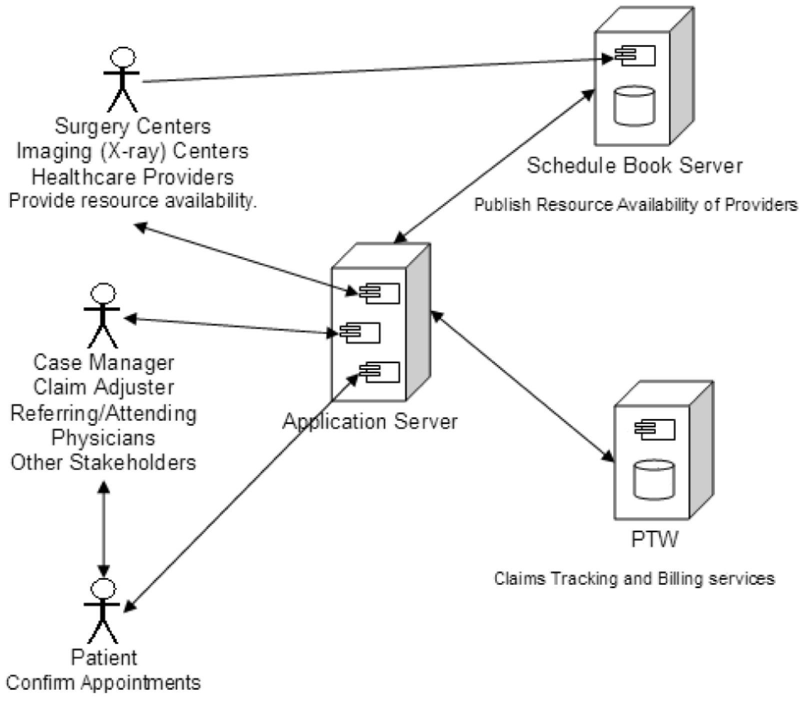

Figure 3. Proposed call center.

Table I. PTW technology changes.

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 e-mail, included with Schedule Server and e-Fax
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 use with COTS

in migrating the PTW to Web services. However, there are also several technical challenges related to such a migration.

4.5. Technical challenges in migrating the PTW to Web services

The primary technical challenge is the risk involved in rebuilding the software system—a risk shared by many large-scale reengineering efforts [41]. Many managers consider cutting-edge technology risky because changes are fast and frequent. From past experience, business has learned a hard lesson regarding adopting new technology that does not create profit.

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. As described in Section 3.1, gaining mastery of such a large collection of new technologies is a non-trivial learning experience. It requires the current PTW maintainers to move from a familiar implementation language and development platform to a more heterogeneous environment with unfamiliar development tools and deployment aids.

The task of rebuilding the system should not be underestimated. The legacy system is written using hand-tuned C code with performance characteristics that are well understood, something that cannot be said with certainty for the Web services solution. Web services introduce a new paradigm in software systems architecture and a multitude of potentially new programming languages.

The prospective solution as presented requires purchasing COTS products to provide key functionality. There are many hidden risks associated with enterprise application integration using COTS. From a business perspective, there is the challenge of managing third-party relationships. This issue encompasses many of the other challenges with respect to our model. Short- and long-term return on investment is affected by marketing prowess and competition. By partnering with a major solution provider, there is the potential to leverage a larger marketing arm, expertise, and technical competence. However, there may be potential conflicts in business priorities. Depending on the level of integration, a critical failure or disagreement in third-party support could be disastrous once the new solution is deployed. Choosing a large solution provider may also affect how the product is implemented and deployed, which COTS products are integrated, and ultimately the business value attained.

From a technical perspective, the COTS products might not support the desired functionality in an exposed API. For example, the schedule server under consideration may 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 desired functionality. For the migration to Web services to be successful, all of these risks must be assessed and mitigated.

5. CONCLUSIONS

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. Skeptics 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 of a truly interoperable and heterogeneous application environment.

At the same time, the challenges in adopting Web services should not be overlooked either. Web services technology is still somewhat immature, with the underlying technologies—XML, SOAP, WSDL, and UDDI—being relatively new. 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 infrastructure issues that must be addressed before Web services become as ubiquitous as other operating systems services. Clearly, more research and practice is needed before migration via Web services will be commonplace [42].

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 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 [43].

Web services need to provide a technical advantage over previous middleware solutions and provide a business advantage for the company over its competitors. Web services 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 benefits accruing even in smaller scale trials, and with results that are clear and definable (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 business value 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 with considerable technical challenges. Even assuming it was accomplished, the learning curve and infrastructural requirements imposed by the introduction of Web services would be considerable.

The real value of Web services has yet to be realized. 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 to improve access to legacy systems, or to develop new ones, the real impact of Web services is still to be felt.

APPENDIX A. THE WEB SERVICES PROTOCOL STACK

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 II.

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 service 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 encountered with earlier middleware technologies, such as CORBA and DCOM [44].

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 [45]. Its use is quite limited when compared with the more common XML messaging mechanism, the Simple Object Access Protocol (SOAP) [46]. 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 is sent using HTTP.

Table II. Web Service Protocol Stack (adapted from [2]).

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)
Network	TCP (Transmission Control Protocol) IP (Internet Protocol)

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 [47]) 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 registering 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 are 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.

ACKNOWLEDGEMENTS

The authors thank the anonymous reviewers of this paper for their insightful comments.

REFERENCES

1. Snell J, Tidwell D, Kulchenko P. *Programming Web Services with SOAP*. O'Reilly & Associates: Sebastopol CA, 2002; 1.
2. Cerami E. *Web Services Essentials*. O'Reilly & Associates: Sebastopol CA, 2002: 3.

3. Port O. The next Web. *Business Week* 2002; **3772**:96–102.
4. The World Wide Web Consortium (W3C). *Web Services*. http://www.w3.org/2002/ws [6 November 2003].
5. Molinaro M. *Gnutelliums—Gnutella Download*. Gnutelliums LLC, Los Angeles CA. http://www.gnutella.wego.com [23 November 2003].
6. Kirda E, Jazayeri M, Kerer C, Schranz M. Experiences in engineering flexible Web services. *IEEE Multimedia* 2001; **8**(1):58–65.
7. Mears J. ASP customers hail return on investment. *Network World* 2002; **19**(12):32.
8. Lambert LJ, Harold B, Hegg PA, Houtz JC, James BJ, Maguire F. Desktop outsourcing: Positive results reported, but analyses could be strengthened. *Report GAO-02-329*, U.S. General Accounting Office, Washington DC, 2002; 68pp. http://www.gao.gov.
9. Zachman J. Enterprise architecture: The issue of the century. *Database Programming and Design* 1997; **10**(3):44–53.
10. Tilley S, Smith D, Wong K. Enterprise integration and Web services. *CASCON 2002 Workshop Reports*. Litoiu M, Lau T (eds.). IBM Centre for Advanced Studies: Toronto, 2002; TR-74.188-31.
11. Finkelstein C. *Technologies for Enterprise Application Integration*. Information Engineering Services Pty Ltd., Perth, Australia. http://members.ozemail.com.au/~ieinfo/ten12.htm [6 November 2003].
12. Smith D. *Enterprise Integration (EI)—SEI Internal Research and Development (IRAD) Project*. Software Engineering Institute, Carnegie Mellon University, Pittsburgh PA. http://www.sei.cmu.edu/plp/EI_IRAD [6 November 2003].
13. Linthicum D. *Enterprise Application Integration*. Addison-Wesley: Reading MA, 1999; 400pp.
14. Wasserman A. Tool integration in software engineering environments. *Proceedings of the International Workshop on Environments*, Chinon, France, 18–20 September 1989. Springer: New York, 1990; 137–149.
15. Brown A, Carney D, Morris E, Smith D, Zarrella P. *Principles of CASE Tool Integration*. Oxford University Press: New York, 1994; 271pp.
16. Brown A, Caldwell W, Long F, Morris E, Zarrella P. Experiences with a federated environment testbed. *Proceedings of the European Software Engineering Conference (ESEC'93)*, Garmisch, Germany, September 1993, Sommerville I, Paul M (eds.). Springer: Berlin, 1993; 175—196.
17. Tilley S. Domain-retargetable reverse engineering II: Personalized user interfaces. *Proceedings International Conference on Software Maintenance (ICSM 1994)*, Victoria BC, 19–23 September 1994. IEEE Computer Society Press: Los Alamitos CA, 1994; 336–342.
18. Smith D, Tilley S. On the role of Net-centric computing in enterprise integration architectures. *Proceedings of the 1st ASERC Workshop on Software Architecture (AWSA 2001)*. Alberta Consortium for Software Engineering Research: Edmonton, Canada, 2001; 24–25.
19. Finkelstein C, Aiken P. *Building Corporate Portals using XML*. McGraw-Hill: New York, 2000; 529pp.
20. Chester T. Cross-platform integration with XML and SOAP. *IT Professional* 2001; **3**(5):26–34.
21. Krill P, Jones M. Intel to bear chip fruit: Web services inspires processor, hardware infrastructure rethink. *InfoWorld* 2002; **24**(14):17–18.
22. Compaq Computer Corporation. *Automating the Web: Compaq's Web Language (WebL)*. http://www.research.compaq.com/SRC/WebL [6 November 2003].
23. Handler J. Agents and the semantic Web. *IEEE Intelligent Systems* 2001; **16**(2):30–37.
24. Coyle F. *XML, Web Services, and the Data Revolution*. Addison-Wesley: Reading MA, 2002; 356pp.
25. Bergey J, O'Brien L, Smith D. Mining existing assets for software product lines. *Report CMU/SEI-2000-TN-008*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh PA, May 2000; 32pp.
26. Barney J. Firm resources and sustained competitive advantage. *Journal of Management* 1991; **17**:99–120.
27. Au Y, Kauffman R. Should we wait? Network externalities, compatibility, and electronic billing adoption. *Journal of Management Information Systems* 2001; **18**(2):47–63.
28. Choi J, Thum M. Market structure and the timing of technology adoption with network externalities. *European Economic Review* 1998; **42**(2):225–244.
29. Katz M, Shapiro C. Network externalities, competition, and compatibility. *The American Economic Review* 1985; **75**(3):424–440.
30. Tilley S, Huang S. On the emergence of the renaissance software engineer. *Proceedings of the 1st International Workshop on Web Site Evolution (WSE'99)*, Atlanta GA, 5 October 1999. University of California: Riverside CA, 1999; 38–41.
31. Microsoft Corporation. *Microsoft .NET*. http://www.microsoft.com/net [26 July 2003].
32. Chikofsky E, Cross J. Reverse engineering and design recovery: A taxonomy. *IEEE Software* 1990; **7**(1):13–17.
33. Müller H, Tilley S, Orgun M, Corrie B, Madhavji N. A reverse engineering environment based on spatial and visual software interconnection models. *Proceedings of the 5th ACM SIGSOFT Symposium on Software Development Environments (SIGSOFT'92/SDE 5)*, Tyson's Corner VA, 9–11 December 1992. Association for Computing Machinery: New York, 1992; 88–98.
34. Tilley S. Net-centric computing and its implications for product lines. *Proceedings of the 10th Annual Software Technology Conference (STC'98)*, Salt Lake City UT. Software Technology Support Center: Hill Air Force Base UT; 1998.

35. Müller H, O'Brien L, Tilley S, Wong K. Report from the 2nd International Workshop on Adoption-Centric Software Engineering (ACSE 2002). *Proceedings of the 10th International Conference on Software Technology and Engineering Practice (STEP 2002)*. IEEE Computer Society Press: Los Alamitos CA, 2003; 74–81.
36. Bergey J, O'Brien L, Smith D. Options analysis for reengineering (OAR): A method for mining legacy Assets. *Report CMU/SEI-2001-TN-013*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh PA, June 2001; 31pp.
37. Ricca F, Tonella P. Understanding and restructuring Web sites with ReWeb. *IEEE Multimedia* 2001; **8**(2):40–51.
38. Empress Software, Inc. *Empress Embedded Database*. http://www.empress.com [6 November 2003].
39. Crystal Decisions, Inc. *Crystal Reports 9 User Guide*. Crystal Decisions, Inc.: Palo Alto CA 2003; 713pp.
40. Adobe Systems, Inc. *Adobe Acrobat Family*. http://www.adobe.com/acrofamily/main.html [6 November 2003].
41. Bergey J, Smith D, Tilley S, Weiderman N, Woods S. Why reengineering projects fail. *Report CMU/SEI-99-TR-010*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh PA, 1999; 37pp.
42. Lavery J, Boldyreff C, Ling B, Allison C. Laying the foundation for Web services over legacy systems. *Proceedings 4th International Workshop on Web Site Evolution (WSE 2002)*. IEEE Computer Society Press: Los Alamitos CA, 2002; 3–12.
43. Rogers E. *The Diffusion of Innovations* (4th edn). Free Press: New York, 1995; 518pp.
44. Cappell D. *Understanding .NET: A Tutorial and Analysis*. Addison-Wesley: Reading MA, 2002; 348pp.
45. UserLand Software, Inc. *XML-RPC Home Page*. http://www.xmlrpc.com [6 November 2003].
46. Rodes K. *XML-RPC versus SOAP*. Masukomi.org, Cambridge MA. http://www.weblog.masukomi.org/writings/xml-rpc_vs_soap.htm [6 November 2003].
47. Microsoft Corporation. *Web Services Discovery Tool (Disco.exe)*. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cptools/html/cpgrfwebservicesdiscoverytooldiscoexe.asp [6 November 2003].

AUTHORS' BIOGRAPHIES

Scott Tilley is an Associate Professor in the Department of Computer Sciences at the Florida Institute of Technology, an Adjunct Associate Professor in the Department of Computer Science & Engineering at the University of California, Riverside, and Principal of S. R. Tilley & Associates, an information technology consultancy based on Florida's Space Coast. He has a PhD from the University of Victoria. He is the General Chair for the *5th International Workshop on Web Site Evolution* (WSE 2003) and the current President of ACM SIGDOC.

John Gerdes is an Assistant Professor in the A. Gary Anderson Graduate School of Management at the University of California, Riverside. His research and teaching interests include Web data mining, distance learning, database systems, e-commerce, and system analysis & design. He has a PhD in Management Information Systems from Vanderbilt University.

Terrance Hamilton is a PhD student in the Department of Computer Science at the University of California, Riverside. His research interests include software engineering, distributed systems, and Web services. He teaches courses on computer networks, object-oriented programming in C++ and Java, and Solaris system administration. He is the lead consultant at IntraMed Corp., a medical services software company.

Shihong Huang is a PhD candidate in the Department of Computer Science at the University of California, Riverside (UCR). Her research interests include reverse engineering for program understanding, software documentation for real-time systems, and multilingual Web sites. She was the primary researcher on a collaborative project related to system redocumentation involving UCR and BMW Group. She is a member of the program committee for WSE 2003.

Hausi Müller is a Professor of Computer Science at the University of Victoria, Canada. He is also a Visiting Scientist with the Centre for Advanced Studies at the IBM Toronto Laboratory, and a Visiting Scientist at Carnegie Mellon University's Software Engineering Institute. He is a principal investigator of CSER. Together with his research group he investigates technologies to build adoption-centric software engineering tools and to migrate legacy software to object-oriented and network-centric platforms. Dr Müller's research interests include software engineering, software evolution, reverse engineering, software reengineering, program understanding, software engineering tool evaluation, and software architecture. He was General Chair for ICSE 2001 and the General Chair for IWPC 2003.

Dennis Smith is a senior member of the technical staff in the Product Line Systems Program at Carnegie Mellon University's Software Engineering Institute (SEI). He is the technical lead in the effort for migrating legacy systems to product lines, and currently heads the SEI's Enterprise Integration project. In this role, he has integrated a number of techniques for modernizing legacy systems from both a technical and business perspective. He is co-author of the book *Principles of CASE Tool Integration* (Oxford University Press, 1994). He has published a wide variety of articles and technical reports, and has given talks and keynotes at a number of conferences and workshops. He is also a co-editor of the IEEE and ISO recommended practice on CASE Adoption. He has been general chair of two international conferences, IWPC'99 and STEP'99. He has a PhD from Princeton University.

Kenny Wong is an Assistant Professor in the Department of Computing Science at the University of Alberta. His main areas of research are software architecture, integration, evolution, and visualization. This research includes conducting case studies, building and using integrated environments for reverse engineering, and exploring a framework for continuous, collaborative program understanding. Current industrial collaborations include IBM and KLOC work. He is a principal investigator of CSER and ASERC. He co-manages a Canadian Foundation for Innovation facility to study distributed software development, with connected, experimental laboratories at the University of Calgary and University of Alberta. He was the Program Co-Chair for IWPC 2003 and is the Program Chair for WSE 2003.