Web services are increasingly becoming an important area of business processing and research for distributed systems. Around 15 years ago, starting with the first move in the form of simple object access protocol (SOAP), Web services have steadily become a reality. In the past 15 years or so, several technological developments have brought the issues of integrating enterprise information systems to each other as they need software functionalities which they themselves do not have: first, the creation of distributed systems which are running over global networks and support distributed functionalities; second, abetted by Internet technologies, the characteristics and strategies of the modern business model have significantly changed; and third, the business competitiveness and complexities have also changed dramatically (Khan & Zhang, 2005). Web services support loosely coupled integrations between various services running on a broad range of technology platforms. As pointed out by Manes (2003), the main difference between the Web services and their previous technologies is that earlier technologies supported only one way of communication which required the installation of the same technology platform and same communication protocols. Web services technology allows easy integration between two applications, but it does not mean that a flawless service is guaranteed.

The basic idea behind Web services is that a specific functionality of software running on one machine of an enterprise is accessible to another machine running at another enterprise using specific protocols over the Internet. Providing seamless access to systems functionality without downloading the software is the main concept behind Web services. Web services represent independent software products that interact with other systems to deliver a software service or functionality. The independence and interactions are essential for the robust integration with other systems.

In a nutshell, three standard technologies are used in Web services: SOAP, universal description, discovery, and integration (UDDI), and Web services description language (WSDL). These three technologies facilitate service level collaboration among software services. SOAP is a lightweight protocol that enables communication across all languages, that is, a piece of code written in one language (e.g., Java) communicates to another piece of code written in a different language (e.g., C++). This flexibility provides the power of interoperability between software systems. UDDI is a registry in where different services advertise their availability to other application services. The services could be easily discovered by other application systems just before they are integrated. WSDL is used to specify the format of the services and their contracts. The service contract describes what will be offered, what procedures will be used, and the use of service (Manes, 2003). The description of a Web service written in WSDL is entered in the UDDI registry. The description includes the name of the service, arguments required to use the service, a briefing of the functionality a Web service offers, and so forth.

In the past, enterprises knew their precise business functions clearly. However, it is not the case in the modern business environment which is highly dynamic. Today, enterprises try to explore new opportunities almost every day, and engage in varieties of functionalities with different enterprises, and are heavily dependent on distributed technologies. Most of the enterprises base themselves on Internet to
increase the functional capabilities of their information systems. In order to keep its business competitive with others, an enterprise has to support a variety of functionalities. It is not feasible for the enterprises to enable their information systems to support all possible functionalities that their businesses need. To provide multiple functionalities, most enterprises are realizing that they need loosely coupled easy-to-integrate technologies on which their application systems will base on. There has been a growing interest in deploying such easy-to-compose software services over the Web to support a variety of systems functionalities to the enterprise requirements.

Through Web services, the Internet is becoming an open global computing platform that facilitates business transactions between independent and heterogeneous systems within and across organizational boundaries. Web services have the ability to be composed with other independent services dynamically in order to achieve high level functionality. These allow business processes to be available over the Internet as well as different applications to communicate with each other using standard Internet. Different types of services ranging from entertainment, finance, healthcare, to high-tech real-time computations are being currently offered by various service providers.

Let us look at Amazon.com, Google, or even Yahoo; most of the functionalities we see in these sites are integrated into the portal of other enterprises’ Web sites, such as weather, travel, maps, Web searches, and so forth (Lefebvre & Cimetiere, 2001). For instance, Amazon.com offers Web services for its product search, cart system, and wish lists. eBay uses Web services such as item search, bidding, and auction creation. FedEx uses Web services that are connected to its package shipping and tracking systems. Each enterprise gets much wider exposure to their business environments using Web services. These services are aimed for direct integration with each other within and across enterprises to form Internet-based systems and perform cross-application transactions (Han, Kowalczyk, & Khan, 2006). The relevant services need to be dynamically discovered and composed with little human intervention to form a collaborative application system. Building applications by dynamically integrating services from different locations certainly supports rapid development. This can be achieved by investing much less than required for procuring an information system.

In Web services, we see a great need and potential to develop a new framework for open, dynamic integration of business services with predicable quality outcomes. With the widespread proliferation of Web services, delivering quality has become a critical challenge for the service providers. Service quality determines the ultimate success of the Web services; it is, therefore obvious that an important selling point of different competing services is the assurance of a certain level of quality.

THE CHALLENGES

Despite the positive predictions, and grand success stories, some major technical challenges exist. Web services have not yet become the universally accepted silver bullet for Web-based business integration. The challenges are how to ensure the overall quality of Web services. In Web services the system needs to satisfy the functional requirements of the target application systems in carrying out a transaction or forming a specific virtual organization under a set agreement. The formation of such a collaborative application system and its continued viability is to be based on the compatibility of the functional as well as quality properties and requirements of individual services and their integrations. It is not enough that only the functional requirements are met. It is equally important to ensure that associated quality properties are also satisfied along with the functional requirements. Let us illustrate this phenomenon with a scenario.
Assume a number of travel related enterprises such as tour operators, hotels, and car rental companies are willing to collaborate their businesses by integrating their computerized information systems to deliver sustained services to their customers in a seamless and improved way. These enterprises deal with a whole range of functions related to their businesses, such as making flight booking, hotel reservation, renting cars, organizing shuttling services from and to airports, and so forth. For example, a tour operator may book hotels and car rental in addition to making flight reservation. In that case, the information systems running at the tour operation need to dynamically integrate to a third party software owned by a different enterprise to book the hotels. The system of tour operator may or may not satisfy the quality requirements of the hotel booking system or vice versa. These quality requirements raise several concerns ranging from security to usability, such as:

- How do we dynamically and automatically verify, check, and ensure before an integration that the quality requirements of two different services are compatible to each other for a viable composition?
- What happens if the hotel reservation Web service modifies its connection parameters or it becomes unavailable after it has been advertised or after serving for a while?
- How can the reliability of the car rental Web service be measured and communicated?
- How do these Web services know about the changing security requirements and privileges of each other?
- What kinds of mechanisms are required for maintaining a seamless collaboration among these Web services?
- How does an application system achieve a predictable response time before using third-party services?
- How do the software engineers test Web services that are provided by different vendors running on remote machines?
- How does the provider of a service test a Web service under all its operating environments with all types of unknown applications systems that it would eventually collaborate?

These are the obvious challenges that need to be addressed. Current service-oriented technologies do not provide the solutions to such obvious requirements. The available technology solutions aiming to simplify integration of different software services through a standard interoperability are not adequately addressing the quality concerns such as security, testability, composability, dependability, maintainability, manageability, deployability, interoperability, and so on. While the concepts of Web services appear to be very attracting to enterprises, satisfying the quality requirements during the service integration and service delivery is still an open challenge. Effectively tackling this challenge represents the next major trend in Web services research, and as a result, it requires new perspectives, new concepts, and new techniques to deal with the quality issues of Web services.

**SEARCHING FOR A SOLUTION**

To address the challenges, the approach that we take in this book is to present a broad perspective on Web services quality. The focus in this book is, therefore, on the technical as well as managerial issues related to quality of Web services. This book provides insights into various quality attributes such as testability, security, dependability, maintainability, usability, composability, and so forth. We understand
that the quality issues such as requirements and assurances for two different Web services cannot be the same, but there are underlying similarities in addressing the issue for most enterprise systems.

The main objective of this book is to focus on advances in Web services quality. This will increase awareness of the importance of quality in a Web services context. It seeks to promote the advances in quality issues related to Web services by exchanging views and ideas within the information technology community. The book emphasizes how important it is to achieve and manage quality for Web services as well as using Web services. We believe that this book will bring much-needed focus to some important aspects of Web services quality, and foster more interest in this pressing topic.

This book features research work as well as experience reports on a wide range of topics related to quality aspects of Web services. It does not intend to initiate an academic debate on the need for Web services; instead, it focuses on how Web services could provide quality achievements in an organization, and how quality Web services could be developed.

REFERENCES


READERSHIP

This book is primarily aimed at information systems researchers as well as practitioners who are interested in nonfunctional aspects of Web services. We assume basic familiarity with the concepts of Web services. To support a variety of readers, most of the chapters are kept relatively self-contained. More specifically, the book will suit:

• Researchers, who like to keep track of the quality issues related to Web services, and who will find the book a rich source of material.
• Practitioners such as systems designers, managers, software architects, and planners responsible for the strategic decision of an enterprise for technology evaluation, or for the software integrators who will find this book in its entirety useful.
• Graduate students who may find enough detail on various research issues in Web services. This book could also serve as reference reading for the senior undergraduate students.
ORGANIZATION OF THE BOOK

The chapters of the book are grouped into four parts corresponding to four broad themes: Section I: Development and Application; Section II: Description and Composition; Section III: Testability and Security; and Section IV: Maintainability and Management.

Chapter I focuses upon some experiences in the development, testing, and deployment of systems built on Web services to support a variety of distributed healthcare applications. The middleware that underpins these applications has been developed to meet the needs of various applications, with these applications being drawn from the areas of healthcare delivery, research, and training. The very nature of the applications that the middleware supports—and their associated data—means that nonfunctional properties such as usability, maintainability, and dependability are key concerns, and necessarily influence the development processes.

Chapter II proposes an approach to provide a unique central console for management of both functional and nonfunctional aspects of Web services. It aims at the development of a framework to provide management features to service providers and clients by supporting management activities all along the lifecycle. The framework allows service providers to consider management activities while developing their Web services. It allows clients to select appropriate Web services using different criteria. Clients also make use of the framework to check if the Web services they are actually using or planning to use are behaving correctly in terms of quality requirements.

Chapter III presents an overview of research and development efforts across several different technical communities aimed at enabling efficient and standardized end-to-end delivery of multimedia content over a service-oriented architecture. It focuses primarily on issues related to quality-of-service specification, measurement, and enforcement for different multimedia distribution applications in this space.

Chapter IV proposes virtualization techniques of Web services. It argues that standard high-availability techniques based on the use of Web servers, business-logic-based caching systems, dynamic binding of Web services by programming the access to SOAP message content from the business logic layer, and other kinds of current open problems can now be handled using a common unique technique.

Chapter V focuses on the problem of the proper definition of nonfunctional properties and methods that may be applied in order to estimate their values. First of all, a reader is familiarized with the concept of nonfunctional properties and different views on the quality of Web services. Then, selected approaches to obtain values of nonfunctional properties are presented. The focus of attention is a Web services profiling that seems to be one of the most promising methods to perform this task. The framework presented in this chapter was implemented and tested within the EU Adaptive Services Grid project.

Chapter VI provides a detailed definition of QoS for Web services (WSs) and the importance of QoS in WS management. An analysis of the requirements that are imposed on the WS description and discovery processes based on QoS is given along with a review of related work in QoS-based WS description and discovery, highlighting the main deficiencies of recent approaches. Then, a complete semantic approach for QoS-based WS description and discovery is presented and analyzed. The chapter concludes by drawing directions for further research.

Chapter VII discusses the current status quo in the area of QoS-aware Web services. The issue is divided into three subissues: description, management, and monitoring. By this discussion, this chapter puts research about dependability in relation with ongoing QoS-related efforts in the Web services domain. Thus it identifies how these efforts can be combined in order to form a dependable Web services platform.
Chapter VIII presents a new procedure towards automating the composition of transactional Web services. This composition procedure does not take into account functional requirements only but also transactional ones based on the acceptable termination states model. The resulting composite Web service is compliant with the consistency requirements expressed by business application designers and its execution can easily be coordinated using the coordination rules provided as an outcome of the approach. An implementation of the theoretical results based on OWL-S and BPEL technologies is further detailed as a proof of concept.

Chapter IX presents a comprehensive logic programming framework designed to support intelligent composition of Web services. The underlying model relies on the modeling of Web services as actions, each described by a logic programming theory. This view allows the use of logic-based planning to address the Web service composition problem, taking advantage of the fact that logic-based planning enables the elegant introduction of a number of extensions and generalizations (e.g., dealing with incomplete knowledge and preferences). The theory describing each Web service is encoded as a logic programming module, and different semantics are allowed within different modules, thus better reflecting the practical use of different service description formalisms and ontologies.

Chapter X proposes a methodology that can be used to enhance the testability of Web services. In general, however, testing Web services is much more challenging than testing normal software applications, not because they are inherently more complex, but because of the limited control and access that users of Web services have over their development and deployment. In order to effectively test Web services under these conditions, special measures and approaches need to be taken to enhance their testability. The chapter argues that right from the early phases of development, the testability of services needs to be taken into account and “designed into” services.

Chapter XI presents the research initiative known as aspect-oriented framework for Web services (AoF4WS). This initiative looks into the role of aspect-oriented programming in enhancing Web services with nonfunctional properties that are orthogonal to the primary functional of Web services, without the need for extensive reprogramming. This is an enhancement to a separation between the functional and nonfunctional aspects of Web services, thereby resulting in easier adaptability and maintainability. The AoF4WS initiative is therefore demonstrated using two projects, namely, security concerns of Web services and self-healing Web services.

Chapter XII proposes an open security framework that is based on the concept of personal identity management. Semantic Web research is seeking to develop expert Web services that are a composition of specialized Web services of multiorganizations. It is argued in the chapter that when all the alternatives are exhausted, the industry will come to the conclusion that only the concept of personal identity management is the only approach to provide true user-centric identity management and give users control over the management of their identities.

Chapter XIII primarily focuses on two aspects: first, the security requirements for Web services and the currently available stack of security mechanisms and frameworks for achieving security at various levels of Web-service implementation; and second, how these could be utilized to build security SLAs, which could be enforced on Web services. Later this chapter suggests a conceptual model to represent these SLA clauses and presents an approach to enact them.

Chapter XIV describes a research study with an objective to explore and describe decision factors related to technology adoption. The study utilized the theories of diffusion of innovations and communities of practice as frameworks and a case study of Web services technology in the digital library (DL) environment to develop an understanding of the decision-making process.

Chapter XV details out the challenges faced in service evolution management and the key activities involved, and their role in SOA quality. Change is the only constant, and this concept holds good for
services too. Service maintenance is the most tedious and longest phase of service lifecycle. The more complex the service, the more difficult it is to maintain it. Service maintenance and service evolution mandate a series of best practices and selective models to apply for better execution and administration.

Chapter XVI aims to examine the extent of Web services usage and quality, applying the balanced scorecard methodology in a small business firm as an exploratory case study. This chapter contributes to guidelines and lessons learned that will inform, educate, and promote small businesses on the importance of maintaining the quality of Web services.

Khaled M. Khan, PhD
Qatar University
February 2008