Service-Oriented Architecture, Web Services, XML and Higher Education

Wai Yin Mok[1], and Joseph Fong[2]

1 Department of Economics and Information Systems, University of Alabama in Huntsville, Huntsville, Alabama, 35899 USA
E-mail: mokw@uah.edu
2 Department of Computer Science, City University of Hong Kong, Hong Kong

Abstract. Service-Oriented Architecture (SOA) and Web Services are now making a tremendous impact on the computing world. Many large enterprises, including universities, are using SOA and Web Services to integrate their legacy systems to provide a single unified "point of data" for users. This paper presents a brief introduction to these exciting technologies, and discusses their impacts on higher education. To make the concepts in this paper concrete, we use a Microsoft Visual Basic .NET example to demonstrate the relationships among SOA, Web Services, and XML and show the XML messages exchanged by various entities in a SOA environment.

Keywords: Service-Oriented Architecture, Web Services, XML

1 Introduction

Higher education, like many large enterprises, suffers from high IT (Information Technology) maintenance cost. Much of this high cost is due to isolated legacy systems that are difficult to work together. Too many times different departments maintain their own applications that do not support integration. Worse yet, multiple versions of the same data are often kept in different databases. Human intervention is too often needed to integrate different applications and to synchronize multiple versions of the same data. To solve this elusive integration problem, this paper presents a promising approach called Service-Oriented Architecture (SOA). SOA is based on the proven object-oriented software design methodology, which has saved millions of dollars in the software industry by developing robust and reusable software components. Web Services, a method of implementing SOA, are also of great importance. Unlike other proprietary approaches to the integration problem, Web Services are based on the open standards of the Internet, which make Web Services likely to succeed. In this paper, we shall show the impacts of these emerging technologies have on large universities.

This paper is organized as follows. Section 2 introduces SOA and its underlying design philosophy. Section 3 discusses Web Services and presents the Microsoft

Joseph Fong, Reggie Kwan, Fu Lee Wang (Eds.): Hybrid Learning: A New Frontier, pp. 22-32.
ICHL 2008, City University of Hong Kong, Hong Kong, August 2008.

Visual Basic .Net example of this paper. In Section 4, we share some success stories of applying SOA to solving IT problems in universities. Section 5 concludes this paper and points out some potential research directions.

2 SOA and Encapsulation

In a nutshell, SOA is a business application design philosophy where integration is the main driving force [4]. In SOA, business functions are created as component services, or just components for short, that can be reused and joined together to improve existing business processes or to create new ones [4, 7]. As Figure 1 shows, these SOA components have well-defined interfaces, which define the operations available in the components and the parameters required to call them. The component interfaces are published, or registered, in a central directory in which components can query the operations provided by other components in the system. All components of a SOA system are connected by a network, which is called an enterprise service bus by some authors.

Fig. 1. A Sample Supply Chain Application [7].

SOA is founded on the proven object-oriented software design principle, although there are differences between object-orientation and SOA [6, 8]. An object is an encapsulation of properties and operations together with a well-defined interface. A SOA component, on the other hand, is a group of software objects that work together to provide a set of well-defined business functions. Alternatively, a SOA component can also be a legacy application together with a proxy component that acts as a middleman between the legacy application and the outside world [3, 2], as shown in Figure 2. Following the principle of encapsulation, the internal mechanisms of an object or a component are not known to the outside world. Other objects or components only know the object's interface or the component's interface.

The major difference between an object and a component is that while there can be more than one object instance of a class (e.g., two stacks or three queues), there is usually only one component that provides a particular specialized set of business functions in a SOA environment. For example, in an enterprise there is only one inventory application and thus only one inventory component that provides its related business functions. Essentially, this is equivalent to say a component and a component instance are the same thing. In other words, it is impossible to instantiate

more than one instance of a component. (In terms of object-orientation, that would amount to say there can only be one object instance of a class, which is untrue.) Consequently, the concepts of inheritance and polymorphism, which are based on the concept of classes, are not applicable to SOA.

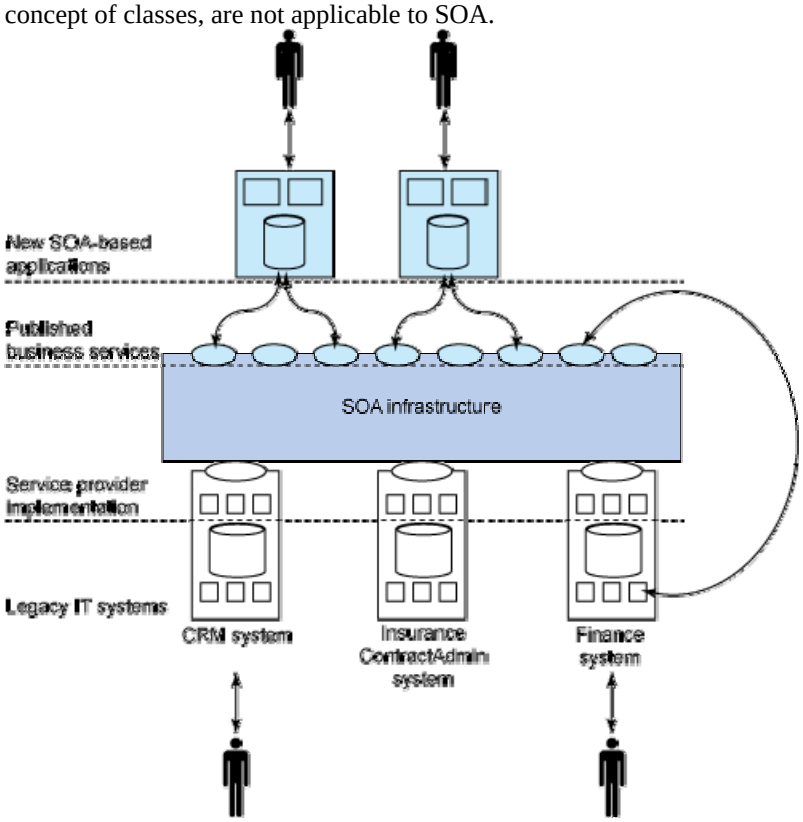

Fig. 2. Legacy IT in SOA Landscape [3].

Because SOA components' interfaces are well-defined and SOA components' internal mechanisms are shielded from the outside world, administrators of a SOA system can then combine and join the SOA components to improve or create business processes, as shown in Figure 3. In a sense, designing new business processes is like putting Lego pieces together, which is the dream of software engineers.

3 Web Services

Web Services are based on the open standards of the Internet. Consequently, Web Services are language and platform independent [9]. Applications using Web Services communicate with each other through HTTP by passing XML messages. As more organizations are connected to the Internet, the concept of applications calling methods over the Internet has become practical. Further, because Web Services

communicate through HTTP, Web Services can get around firewalls, and thus avoiding being blocked. Because of these advantages, Web Services have become a common way to implement SOA [7].

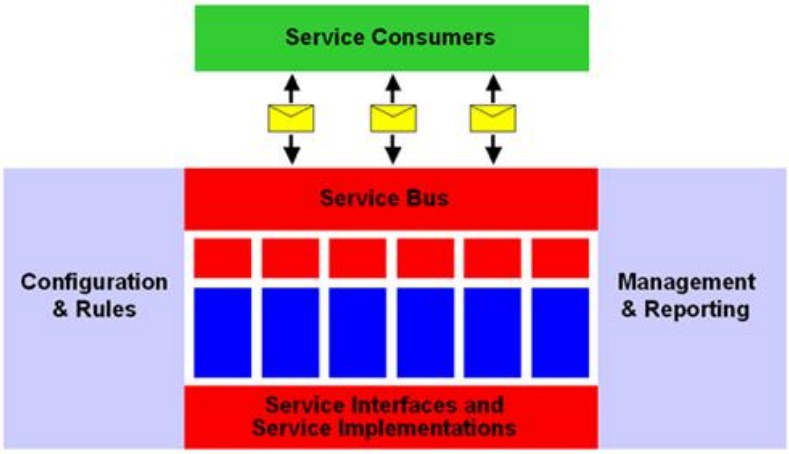

Fig. 3. A Sample Service Architecture [7].

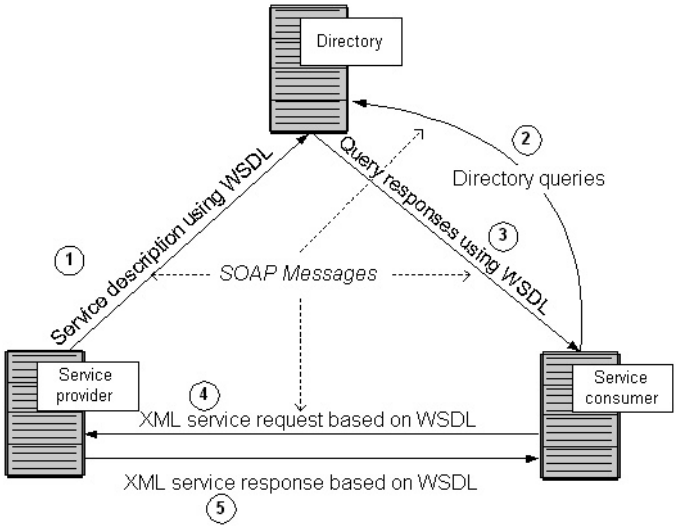

Fig. 4. Web Service Architecture [1].

W3C defines Web Services as "a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks. Web services are characterized by their great interoperability and extensibility, as well as their machine-processable descriptions thanks to the use of XML. They can be combined in a loosely coupled way in order to achieve complex operations. Programs providing simple services can interact with each other in order to deliver sophisticated

added-value services [10].” Behind this definition actually is a simple architecture, which is depicted in Figure 4.

In SOA’s context, the Service Consumer and Service Provider in Figure 4 are components we addressed in Section 2. Whenever a Service Consumer wants a service, it queries the Service Broker for the providers that provide the desired service. The network that connects all components together is simply the Internet, or a private network that employs the Internet protocols. Calling of services is achieved by passing messages, encoded in XML. Figure 4 also shows two fundamental elements of Web Services: SOAP (Simple Object Access Protocol), and WSDL (Web Services Description Language) [9]. In the following, we use an example written in Microsoft Visual Basic .NET to demonstrate the concepts in Figure 4.

Microsoft Visual Basic .NET implements a Web Service as a class. Figure 5 is an example. It defines a Web Service in two files: Service1.asmx and Service1.asmx.vb. These two files together define the methods available in the Web Service, whose purpose is to convert a score in between of 0 and 100 to a letter grade. (Note the reference from the Service1.asmx file to its code-behind file Service1.asmx.vb.) To make this Web Service available to the outside world, we must deploy it to a Web server such as an Internet Information Services (IIS) Web server. However, for this presentation, we simply use the built-in test server of Visual Basic .NET. Methods in a Web Service are invoked through a Remote Procedure Call (RPC). These methods, which are marked with the WebMethod attribute, are in general referred to as Web methods. For example, LetterGrade is a Web method of the Web Service of Figure 5.

```
File Name: Service1.asmx
<%@ WebService Language="VB" CodeBehind="Service1.asmx.vb" Class="LetterGrade.Service1" %
>
-------------------------------------------------------------------------------------------
File Name: Service1.asmx.vb
Imports System.Web.Services
Imports System.Web.Services.Protocols
Imports System.ComponentModel
<System.Web.Services.WebService(Namespace:="http://www.cs.cityu.edu.hk/~ichl2008/")> _
<System.Web.Services.WebServiceBinding(ConformsTo:=WsiProfiles.BasicProfile1_1)> _
<ToolboxItem(False)> _
Public Class Service1
Inherits System.Web.Services.WebService
<WebMethod()> _
Public Function LetterGrade(ByVal score As Integer) As String
If score >= 90 Then
Return "A"
ElseIf score >= 80 Then
Return "B"
ElseIf score >= 70 Then
Return "C"
ElseIf score >= 60 Then
Return "D"
Else
Return "Fail"
End If
End Function
End Class
```

Fig. 5. A Sample Visual Basic .NET Web Service Code.

The Web Service of Figure 5 also publishes a WSDL file, which is an XML document. Part of this file is shown in Figure 6. The purpose of this file is to provide a description of the Web Service. When a client application requests the Web Service’s WSDL description, ASP.NET generates this file and returns it to the client

application. Then, the client will know the Web methods provided by the Web Service and the parameters required to call them.

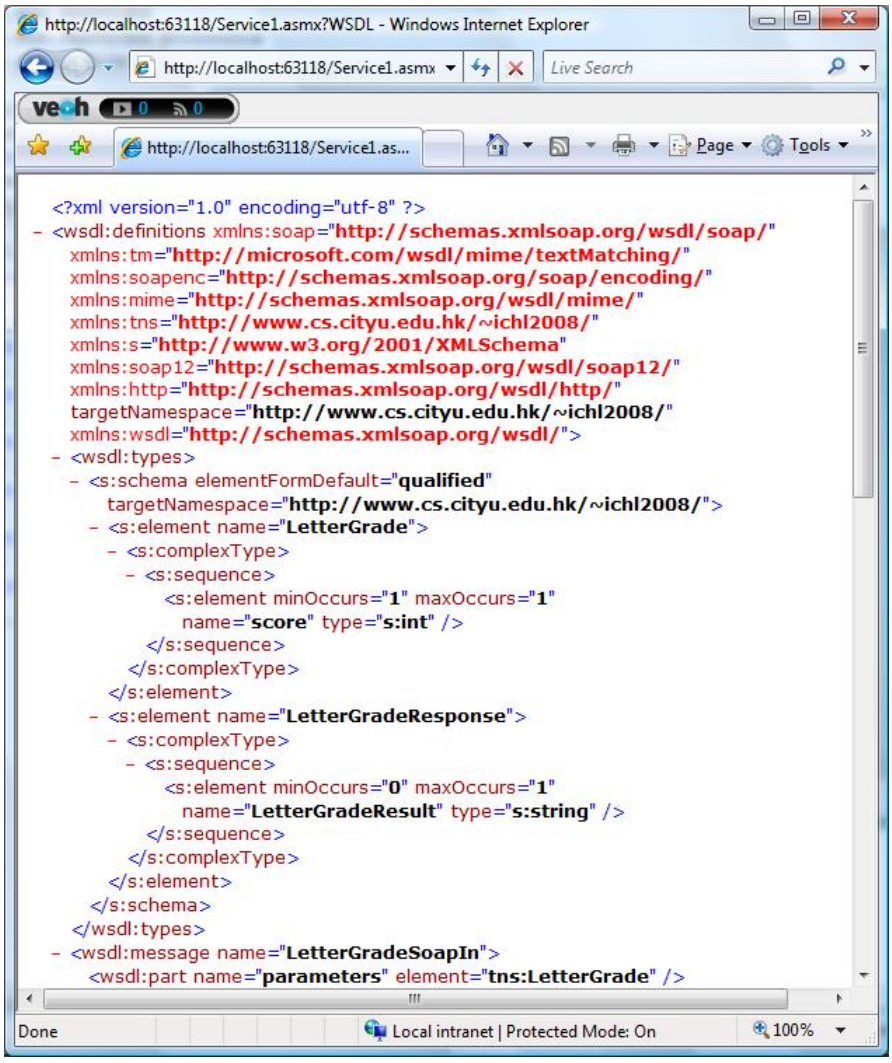

Fig. 6. The WSDL file of the Web Service of Figure 5.

To show an example on how to invoke the Web methods of a Web Service, suppose we have a client application that converts a score in between of 0 and 100 to a letter grade. Such an application will send a SOAP message, which is an XML document, to the Web Service of Figure 5, as described in Figure 4. That SOAP message is displayed in Figure 7. After calculating the letter grade, the Web Service of Figure 5 will respond to the client application by sending it another SOAP

message, which is not shown in this paper because of space limitation. However, it is similar to the XML document in Figure 7.

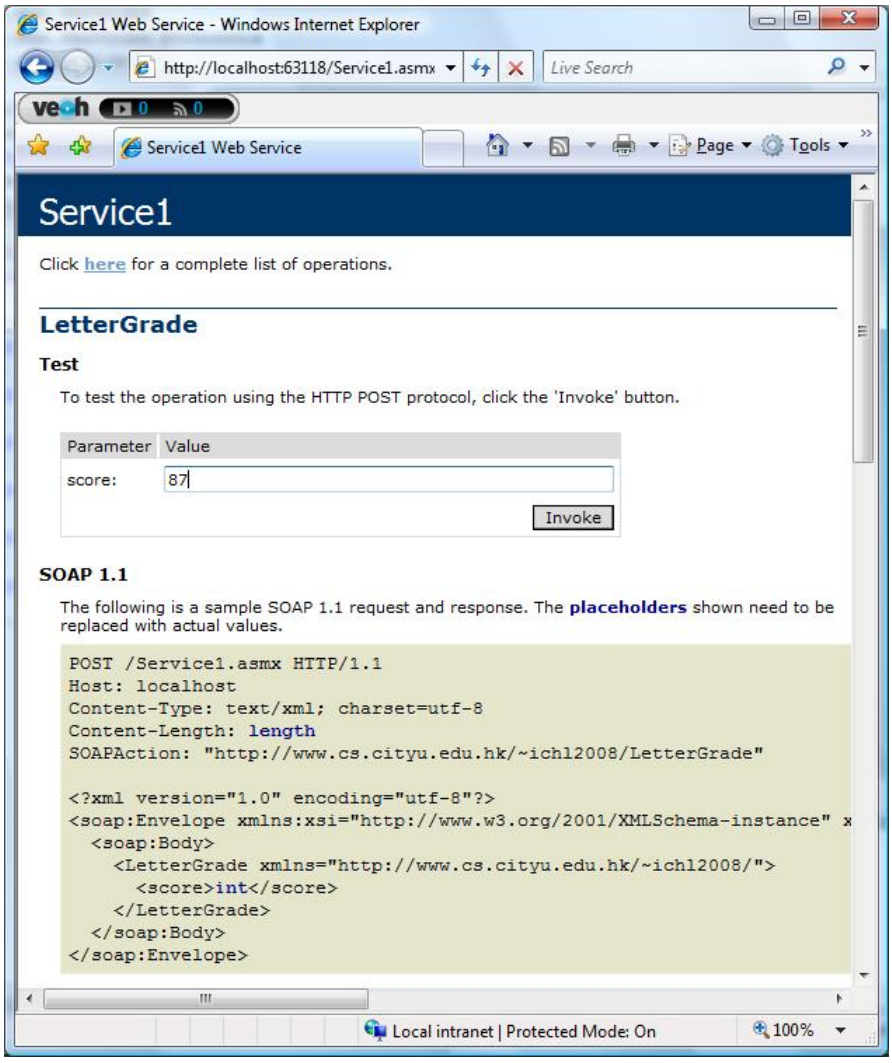

Fig. 7. The SOAP file of the Web Service of Figure 5.

Figure 8 contains the code of a sample client application that invokes the Web methods of the Web Service of Figure 5. In Figure 8, Microsoft Visual Basic .NET first creates a proxy class Service1 based on the WSDL information in Figure 6. This proxy class stands as a middleman between the client application and the remote Web Service. Then, it creates an object instance remoteLetterGrade of this proxy class. Calling the Web methods are done through this object (e.g., calling the method LetterGrade of the object remoteLetterGrade in Figure 8.) The form in Figure 9

appears when we execute the code in Figure 8. Typing in a score and clicking on the button results in the message box in Figure 10, which contains the letter grade “A” if we type in a score of 92, as in Figure 9.

```
File Name: WebServicesTest.vb
Public Class WebServicesTest
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
_
Handles btnWebServicesTest.Click
Dim remoteLetterGrade As localhost.Service1
Dim letterGrade As String
remoteLetterGrade = New localhost.Service1
letterGrade = remoteLetterGrade.LetterGrade(TextBox1.Text)
MessageBox.Show("Grade returned from the Web Service: " & letterGrade)
End Sub
End Class
```

Fig. 8. A Sample Window Client Application Code.

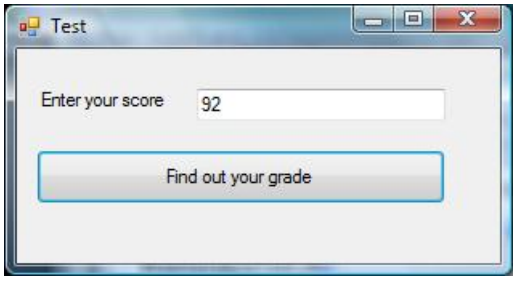

Fig. 9. The Form of the Window Client Application of Figure 8.

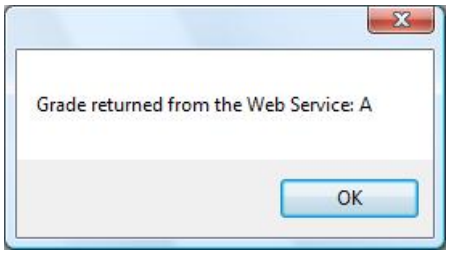

Fig. 10. The Response of the Web Service of Figure 5.

4 Applying SOA in Higher Education

Traditionally, universities have been operated as highly decentralized enterprises, with faculty and business units allowed considerable autonomy to choose their computing systems, business rules, and operating practices. As a result, university-wide assets (e.g., the brand) and operating budgets may face compromises not necessary in a more centrally run environment.

In such an environment, university IT managers may find themselves supporting, at relatively high cost, a diverse array of computing platforms and applications, each with its own programming language, tools, and training requirements. Many institutions of higher education today are saddled with aging legacy systems that are hard to integrate because individual departments were allowed to order machines and applications on an ad hoc basis, with little centralized control.

Burdened with obsolete systems, colleges find themselves having to devote significant time and money to managing multiple interfaces and communication protocols, solely for the purpose of making sure their institution's systems can talk to one another. Too often, users - including students, faculty, administrators, and members of the university community outside of the campus – find that data is inaccurate, inconsistent with information from another database, or too old to meet current requirements. Facing such a scenario, many college senior administrators have characterized themselves as "data rich, but information poor."

In this section, we summarize the findings in [5], which shows how SOA can help solve the IT problems in three colleges.

4.1 The University of Wisconsin at Madison

The University of Wisconsin needed to regain control over its enterprise data to ensure that campus users were working with accurate versions, getting the information they needed, and only the official data they were authorized to receive. As a result, the University decided to replace the existing enterprise systems infrastructure with a service-oriented architecture (SOA).

Once web services are rolled out, enterprise-wide business rules can be standardized, thereby eliminating the "data-misinterpretation" problem. Data about students and employees and their respective roles will be called up from a master registry - the system of record or the "single source of truth." In addition, decisions can be expedited because accurate information is served up dynamically, in real time. For example, if a student shows up at a recreational facility, a staff person can enter the student's name and ID number and know immediately if the student is eligible to use the facility - as a "currently enrolled" student based on uniform, university-wide business rules.

4.2 Embry-Riddle Aeronautical University

Embry-Riddle's leadership saw web-based enterprise systems as the solution to building better business processes for its globe-spanning "extended campus." With so many faculty and students working remotely, Embry-Riddle needed a robust, highly agile, and scalable enterprise resource planning (ERP) system that could provide its globally dispersed staff and students with real-time, web-accessible services, while also improving the speed and accuracy of its business processes.

The bottom-line payoff from the implementation has been that faculty and staff can accomplish key tasks faster and with fewer difficulties. Paper-based workflows that formerly required frequent manual interventions have been transformed into digitized

workflows that significantly cut the time required to complete a process, while also delivering overhead cost savings.

Web services' benefits extend well beyond improved ease of use for students, alumni, faculty, and staff. For the IT department, it means a sizeable reduction in account administration costs. "With single sign-on, we anticipate a 30% savings on IT staff time, because account administration will be vastly more efficient," Becky Vasquez, Director, IT Services, Embry-Riddle Aeronautical University, explains.

4.3 Cornell University

Cornell's Office of Information Technologies faced a problem. Over the years, multiple, independent systems were allowed to develop throughout the University because local units had broad latitude to invest in siloed applications, with little regard to the implications for enterprisewide integration. As a result, at Cornell, administrative computing alone encompasses three major database applications, four operating systems, four hardware platforms, and six development centers.

Cornell's senior IT management refers to the University's current SOA initiative as a business and not a technology architecture, as the strategic driver behind Cornell's enterprise systems rebuild is business process improvement. When coupled with web services, Cornell's new business-driven SOA will allow users to view enterprise data dynamically across silos, and to access componentized applications as needed.

5 Conclusions

This paper presented a brief introduction to SOA and Web Services and used a Microsoft Visual Basic .NET example to explain how a client application invokes the Web methods of a Web Service. We also discussed how SOA can help solve the IT problems in universities.

Some possible research directions in SOA environments are as follows:

1. Security: How can we make a SOA system secure, assuming the SOA components are passing around sensitive materials?
2. Analysis and Design: Given an enterprise, perhaps a university, how can we deploy a SOA system?
3. User Interaction: How can we balance the needs of all users in a SOA environment?

It is certain that many interesting research problems will appear as we investigate these problems.

Acknowledgements:

This paper is funded by CityU Teaching Development Grant 6000152 of City University of Hong Kong.

References

1. Doug Barry. Web Services explained. http://www.service-architecture.com/web-services/articles/web services explained.html.
2. Shantanu Bhattacharya. Integrate legacy systems into your SOA initiative. December 2007. http://www.ibm.com/developerworks/webservices/library/ws-soa-legacyapps/index.html.
3. Jeremy Caine and Joe Hardman. Design strategies for legacy system involvement in SOA solutions. April 2007. http://www.ibm.com/developerworks/webservices/library/ws-soa-legacy/?S TACT=105AGX04&S CMP=ART.
4. Patrick F. Carey and Bernard W. Gleason. Solving the Integration Issue Service-Oriented Architecture (SOA). May 2005. http://t1d.www-03.cacheibm.com/industries/education/doc/content/bin/Service-OrientedArchitecture.pdf.
5. Eduventures. Service-Oriented Architecture and Web Services: The Next Big Thing in University Enterprise Computing. February 2006. http://www.oracle.com/industries/education/eduventures-service-oriented-architecture-and-web-services.pdf.
6. Thomas Erl. Service-Orientation and Object-Orientation Part I: A Comparison of Goals and Concepts. February 2008. http://www.soamag.com/I15/0208-4.asp.
7. Raghu R. Kodali. What is service-oriented architecture? June 2005. http://www.javaworld.com/javaworld/jw-06-2005/jw-0613-soa.html.
8. Boris Lublinsky. Defining SOA as an architectural style. January 2007. http://www.ibm.com/developerworks/architecture/library/ar-soastyle/.
9. W3 Schools. Web Services Tutorial. http://www.w3schools.com/webservices/default.asp.
10. W3C. Web Services Activity Statement. 2002. http://www.w3.org/2002/ws/Activity.